UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/574,639	05/18/2000	Phil Rich	Q00-1057-US1	8105

7590 02/24/2005

TEJPAL S. HANSRA
REGISTRATION NO. 38,172
3705 CANTERBURY LANE, #6
BELLINGHAM, WA 98225

EXAMINER	
DESTA, ELIAS	
ART UNIT	**PAPER NUMBER**
2857	

DATE MAILED: 02/24/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/574,639	RICH, PHIL	
	Examiner	Art Unit	
	Elias Desta	2857	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 2 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *18 May 2000*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☒ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *1-31 and 36-40* is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☒ Claim(s) *32-35* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Detailed Action

Abstract

1. The abstract is objected to because of the following minor informality:

   - Page 40, lines 7 and 9: the phrase "said data" is indefinite. Use the actual phrase, such as "transferring data" or "required data" wherever is appropriate. Avoid the word "said" in the abstract for better clarity.

   Correction is required.

Specification

2. The specification is objected to because of the following minor informalities:

   - Page 2, lines 5-16: the paragraph needs to be revised for clarity and better readability. For instance, the sentence on lines 9-11 is not clear and comprehendible.

The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objection

3. Claims 32-35 are objected to because of the following minor informalities:

With regard to claim 32: the claim cannot depend on itself, hence 33-35 are objected to as being dependent upon a objected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Appropriate dependency and corrections are required.

Allowance

4. Claims 1-31 and 36-40 are allowed. The following is an examiner's statement of reasons for allowance:

In reference to claims 1, 23 and 37: Hanson (U.S. Patent 6,378,013) teaches a system for assessing performance of a device, such as a hard drive (see Hanson, Fig. 3 and column 2, lines 21-44). The method includes acts of establishing a virtual drive. The processor is configured to transfer data between the virtual drive to the hard drive subsystem, and measuring the transfer rate of the subsystem. Another embodiment of the system includes a program storage device storing instruction when executed by the computer perform a method of measuring the performance of the hard drive in a computer.

However, Hanson does not teach that the system specifies more or more different required data transfer rate. The claimed invention further includes a method that measures the actual data transfer rate for each access pattern and determines performance of the storage device in relation to at least one required data transfer rate as a function of the required data transfer rate and the actual data transfer time of data for a given access pattern.

Citation of pertinent prior art:

- *Srikrishna et al.* (IEEE Article, 'Predicting Track Mis-registration (TMR) From Disk Vibration of Alternate Substrate Material') teaches a method of quantitative measure for closed loop TMR due to disk vibration.
- *Pentakalos et al* (IEEE Article, 'Analytical Performance Modeling of Hierarchical Mass Storage Systems') teaches a queuing network model that can be used to carry out capacity planning studies.
- *Getreuer* (U.S. Patent 6,741,529) teaches method and apparatus for moving carriage assembly from initial position to target position and optical disc system.
- *Smith et al.* (U.S. Patent 6,546,456) teaches method and apparatus for operating vehicle mounted disk drive storage device.
- *Klein* (U.S. Patent 5,951,700) teaches method of computer system usage determination based on hard disk drive activity.
- *Liu* (U.S. Patent 5,768,617) teaches intelligent hardware for automatically reading and writing multiple sectors of data between a computer bus and a disk drive.
- *Shimizu et al.* (U.S. Patent 5,383,068) teaches head position recognition method, a speed calculation method, and a head movement speed control device.

The remaining claims are dependent upon *claims 1, 23 and 27* and contain further limitations.

Conclusion

5. This application is in condition for allowance except for the objections to the abstract, specification and claims as noted above.

Prosecution on the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

A shortened statutory period for reply to this action is set to expire TWO MONTHS from the mailing date of this letter.

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Elias Desta whose telephone number is (571)-272-2214. The examiner can normally be reached on M-Thu (8:30-7:00).

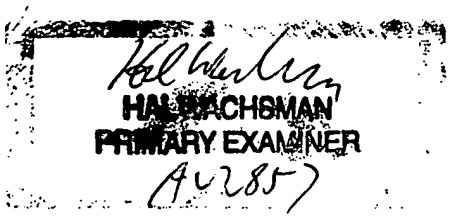
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Marc S. Hoff can be reached on (571)-272-2216. The fax phone numbers for the organization where this application or proceeding is assigned are (703)-308-5841 for regular communications and (703)-308-5841 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)-308-1782.

Elias Desta
Examiner
Art Unit 2857

-ed

February 16, 2005



Notice of References Cited	Application/Control No. 09/574,639	Applicant(s)/Patent Under Reexamination RICH, PHIL	
	Examiner Elias Desta	Art Unit 2857	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,741,529	05-2004	Getreuer, Kurt W.	369/30.17
	B	US-6,546,456	04-2003	Smith et al.	711/112
	C	US-5,951,700	09-1999	Klein, Dean A.	714/47
	D	US-5,768,617	06-1998	Liu, Andy J.	710/5
	E	US-5,383,068	01-1995	Shimizu et al.	360/78.06
	F	US-6,378,013	04-2002	Hanson, James H.	710/100
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Srikrishna et al., 'Predicting Track Misregistration (TMR) from Disk Vibration of Alternate Substrate Materials', January 2000, IEEE Article, Volume 36, No. 1, pages 171-176
	V	Pentakalos et al., 'Analytical Performance Modeling of Hierarchical Mass Storage Systems', October 1997, IEEE Article, Vol. 46, No. 10, pages 1103-1118
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Predicting Track Misregistration (TMR) From Disk Vibration of Alternate Substrate Materials

Padmanabhan Srikrishna and Kumar Kasetty

Abstract—**Disk vibration is a significant contributor to track mis-registration (TMR) in todays high performance drives with tracks per inch (TPI) growing at the rate of more than 60% per year and disk rotating speeds of 10 000 rpm. Aluminum disk substrates result in significantly large TMR that can limit the TPI that can be achieved and hence the drive industry is being forced to consider alternate disk substrate materials. In this paper we identified a quantitative measure called "displacement metric" that predicted closely the TMR due to vibration of disk substrate materials. The displacement metric depends on substrate material properties of Young's Modulus, damping and Poisson's ratio and is a better predictor of TMR than traditional measures like specific stiffness. Many substrate vendors are now using this metric to improve the performance of new alternate substrate materials. Laser doppler vibrometer (LDV) measurements in a 10 000-RPM disk drive with aluminum, glass, glass-ceramic, alumina, silicon carbide and zirconia disks showed that the cumulative 3-sigma TMR metric due to disk vibration was proportional to the reciprocal of the "displacement metric." Furthermore, we developed a methodology to predict the closed loop TMR due to disk vibration. The closed-loop TMR due to disk vibration was compared for a range of servo bandwidths (800–1600 Hz) for aluminum and alternate substrate materials. This modeling work has proven to be useful in the development of new disk substrate materials and predicting TMR due to disk vibration for higher performance disk drives.**

Index Terms—**Alternate disk substrates, disk vibration, track misregistration.**

I. Introduction

DEMAND for higher data storage capacity in hard drives has led to ever increasing levels of data density on disk platters. Current drive designs operating at 10 000 rpm employ high values of Tracks per inch (TPI) to achieve this goal. Inside the drive, air turbulence generated by the rotating disk stack is responsible for inducing vibration in both the disks and actuator arm. Due to these combined effects, track misregistration (TMR) during read and write operations can reach unacceptable levels. Disk vibration is a significant contributor to track misregistration (TMR). In the past, aluminum disk substrates have met the TMR requirements. However, with much higher TPI and disk rotating speeds now, the drive industry is being forced to consider alternate disk substrate materials. Understanding of the material properties that contribute to disk vibration is important in the development of higher performance disk substrates.

In this paper we have identified a quantitative measure called "Displacement Metric" that depends only on the substrate material properties of Youngs Modulus, damping and Poisson's ratio. The displacement metric must be maximized in order to reduce disk vibration. The damping ratio in this metric was estimated from the Quality (Q)-factor corresponding to the fundamental frequency in the displacement spectrum of the disk. We conducted laser doppler vibrometer (LDV) measurements of disks made of aluminum and alternate substrate materials in a 10 000-rpm disk drive. The substrate materials studied were aluminum, glass, glass-ceramic, alumina, silicon carbide and zirconia disks. The measured axial displacements were converted to radial track misregistration (TMR) in % data track units. The area beneath the power spectral density was computed to determine the cumulative 3-sigma TMR metric due to disk vibration. Our results indicated that the reciprocal of the "displacement metric" closely matched this open loop TMR metric due to disk vibration.

Disk vibration is one of the sources of position-error during read and write operations in a drive. Previous work has failed to consider how performance of disk substrate materials is altered within a closed loop system for a set of actuator plant dynamics and servo compensator design. This paper models the disk vibration as a disturbance to closed loop system and determines the closed loop TMR for a series of servo compensators with varying servo bandwidths. The closed loop TMR is compared for aluminum and alternate substrate materials.

II. Reduction of Disk Vibration: Theory

This section develops a theoretical model to understand the various factors that contribute to disk vibration in a drive. For given disk dimensions, this model determines the substrate material properties that influence disk vibration.

Each vibration mode of disk has m nodal circles and n nodal diameters and is designated by (m, n). An approximate relationship between disk vibration amplitude and substrate material properties and geometry is found from the elastic theory of thin annular disks [1]. The natural frequency ω of an annular disk that is clamped at the inner diameter and free at the outer diameter is:

$$\omega_{mn} = \frac{\lambda^2 h}{2\pi a^2}\sqrt{\frac{1}{12(1-\nu^2)}}\sqrt{\frac{E}{\rho}} \qquad (1)$$

Manuscript received July 16, 1999. This work was completed using 10 000 rpm preproduction disk drives at Quantum Corporation. Saint Gobain Industrial Ceramics provided samples of silicon carbide, alumina, and zirconia disk substrate materials.

The authors are with the Advanced Technology Group, Quantum Corporation, Shrewsbury, MA (e-mail: srikrish@alum.mit.edu; kumar.kasetty@quantum.com).

Publisher Item Identifier S 0018-9464(00)00263-6.

TABLE I
MODAL PARAMETER "LEMDA-SQUARE" FOR DIFFERENT VIBRATION MODES OF ANNULAR DISK FROM PLATE THEORY [1]

Radial Mode	Circum. Mode	Dimensionless Parameter lemda-square			
(m)	(n)	b/a=0.1	b/a=0.3	b/a=0.5	b/a=0.7
0	0	4.23	6.66	13	37
0	1	3.14	6.33	13.33	37.5
0	2	5.62	7.95	14.7	39.3
0	3	12.4	13.3	18.5	42.6

where

Young's Modulus	E
Density	ρ
Poisson's Ratio	ν
Disk outer radius	a
Disk clamping radius	b
Disk thickness	h
Mode	(m, n)
Dimensionless modal parameter (lemda-square)	λ_{mn}^2, (refer Table I)

The maximum vibration amplitude (X) is approximated by a 2nd order, 1-D, mass-spring-damper system [4] as

$$X = \frac{F_0}{k\left[(1-(\omega/\omega_n)^2)^2 + (2\xi\omega/\omega_n)^2\right]^{1/2}} \tag{2}$$

where,

ω_n undamped natural frequency = $\sqrt{k/m}$

m = disk mass = $\rho\pi(a^2 - b^2)h$

ξ damping ratio.

Thus at resonance, the maximum displacement is

$$X_r = \frac{F_0}{2\xi\omega_n^2 m}. \tag{3}$$

Replacing the frequency term from (1), we get (4), which can be written as a multiple of two terms.

$$X_r \propto \left[\frac{a^2}{h^3\lambda^4}\right] \cdot \left[\frac{(1-\upsilon^2)}{E\xi}\right]. \tag{4}$$

Thus the vibration amplitude is decreased by

1) By modifying disk dimensions and clamping conditions, i.e., by reducing disk radius (a); increasing disk thickness (h) or increasing the clamping diameter (b) to increase the lemda-square parameter.
2) By using a substrate material with larger Young's Modulus (E), damping ratio (ξ) and Poisson's ratio. These parameters form the displacement metric expressed in the equation below. The displacement metric must be increased to decrease vibration amplitude.

$$\text{Displacement Metric} = \frac{E\xi}{(1-\upsilon^2)}. \tag{5}$$

TABLE II
THE PROPERTIES OF THE DISK SUBSTRATE MATERIALS

	Aluminum	Glass	Glass ceramic	Zirconia	Alumina	Silicon Carbide
Young's Modulus(Gpa)	71	100	123	223	400	430
Density d (kg/m^3)	2660	2760	2700	6040	3900	3210
Poisson's Ratio	0.30	0.23	0.24	0.25	0.25	0.18

The displacement metric is used in this paper to predict and compare the performance of substrate materials having the same disk dimensions. The damping parameter in (5) is dependent on the material properties and the fluid dynamics that provide turbulent excitation to the rotating disk. In a previous publication [5], the authors had described the evaluation of damping of alternate substrates using transient response method and Q-factor methods [4] and shown that the damping parameter determined from Q-factor method matched experimental results more closely.

III. EXPERIMENTAL METHODS

Velocity spectrum measurements were made in three drives running at a constant speed of 10 000 rpm using a Polytec fiber-optic laser doppler vibrometer (LDV) with a laser beam passing through a slot in the top cover to measure velocity at the OD of the top disk. The disk substrates tested had an outer diameter of 84 mm and inner diameter of 25 mm and thickness of 0.8 mm (31.5 mil). Flatness was within 10 microns and surface roughness less than 100 Å. Samples of aluminum, glass, glass ceramic, zirconia, alumina and silicon carbide disks were tested (Table II lists the properties). The average results from three tests were summarized for comparison purposes.

The experiments were performed with the disk drive clamped to avoid any baseplate motion and to perform repeatable experiments. The laser beam was positioned using the micrometers for the laser beam to reflect directly back into the lens probe. The lens probe was focussed for good signal to noise ratio. The velocity output from the Vibrometer controller was provided as an input to a Hewlett-Packard (HP) 35 670 dynamic signal analyzer. We used the highest sensitivity, 25 mm/s/V, of the controller with the measured signal without saturation and used a 100 kHz-tracking filter. The HP analyzer computed the power spectrum of velocity in (volts rms) with a flattop window with 50 averages, for a span of 0–3200 Hz. The disk harmonics (166.67*n Hz for 10 000 rpm) were removed to focus our attention on the disk resonant modes alone. The displacement spectrum was determined by integrating the velocity data and removing the low frequency terms associated with integration noise. The axial displacements were converted to radial track misregistration units using a modal analysis study of the disks. As Fig. 1 shows, the track misregistration is proportional to the slope of the disk at a given radius. The FEM analysis determined the slope/displacement ratio to be 0.051. The factor [f = 0.051* (Disk thickness + 0.5*Slider thickness)] was multiplied by the axial displacement in order to obtain the radial TMR. The radial displacement was expressed in percentage data track width by dividing by data track width (m) and multiplying by 100. For this study we chose a TPI of 25 000 that translates to a track width of 1.016e-06 m. The power spectral density (m^2/Hz) was determined from the displacement spectra and the

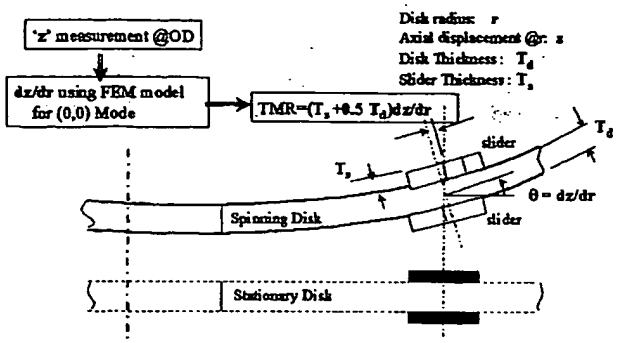


Fig. 1. Computing track misregistration (TMR) from axial displacement (z) at a given radius (r).

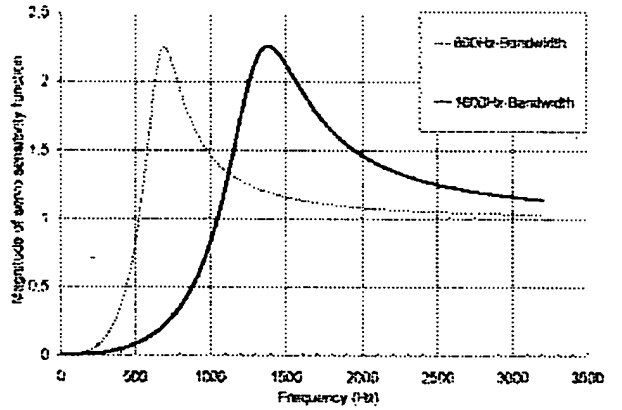


Fig. 2. Low (800 Hz) and high (1600 Hz) servo bandwidth model servo sensitivity curves show different responses of the servo system to disk vibration.

total area under the trace represents the variance, σ^2, of the TMR [2]. The standard deviation given by the square root of the variance, and three standard deviations represents the 0 to peak vibration assuming a Gaussian distribution. The peak cumulative 3σ TMR was determined for each substrate which represents the open-loop contribution of disk vibration to TMR.

IV. Closed Loop TMR Prediction for Alternate Substrates

During read and write seek operations in a disk drive, disk vibration is one of the sources of disturbances that cause position error due to track misregistration (TMR). Actuator torque disturbance, actuator dynamics, measurement noise, bearing defects and spindle harmonics influence this TMR also. A simple model of the control system was used to generate the servo sensitivity function magnitude, which affected how disk vibration and other position disturbances influence the position error. The plant model used was $1/(J*s^2)$ and a compensation model $K*(s^2/w^2+2*\zeta/w*s+1)/s$, resulted in a stable realizable loop transfer function with the general shape of a realistic control system. The servo bandwidth and phase margin are typically limited by sampling effects, the system dynamics (resonances), and the desire for robustness to variations across heads in a given drives and across drives in a mass production run of a given drive type. We varied the bandwidth or the open-loop crossover frequency from 800–1600 Hz to understand the TMR contribution

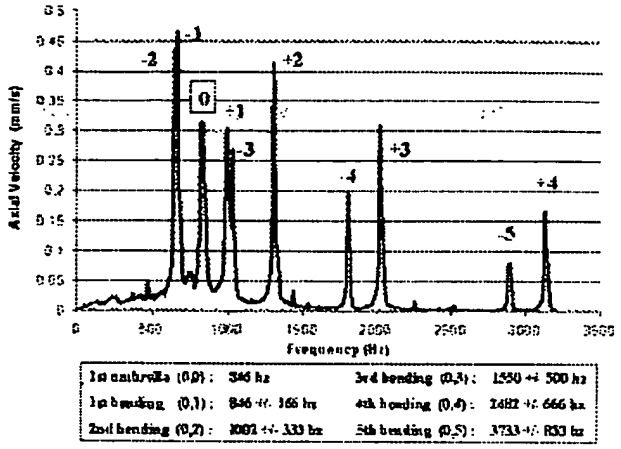


Fig. 3. The resonant modes of disk vibration for an 84 mm OD and 0.8 mm thick aluminum disk spinning at 10 krpm.

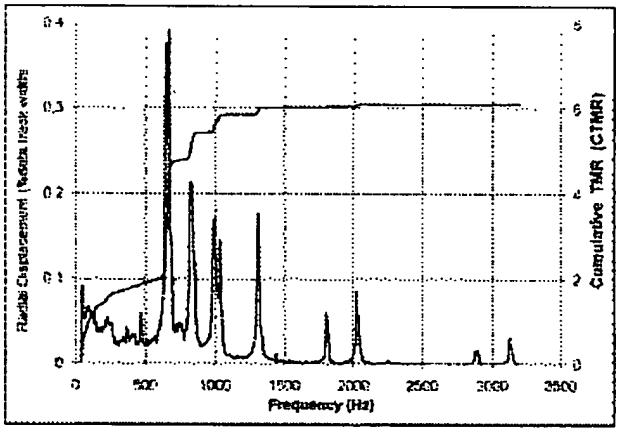


Fig. 4. The disk vibration spectra (in displacement units) and the cumulative TMR without servo weighting (6.3% data track width) for aluminum disk.

from different disk substrate materials in a closed-loop system. We assumed a desired phase margin of 30° (usually between 10 and 90) and the zero damping, $\zeta = 0.7$ (typically 0.5–4). The zero damping affected servo speed of response and the low frequency rejection characteristics of the sensitivity function. The low frequency gain K of the compensator was adjusted to achieve the desired loop crossover.

The servo sensitivity function is plotted for a servo bandwidth of 800 and 1600 Hz in Fig. 2. The low (800 Hz) bandwidth sensitivity function attenuates disturbances like disk vibration below 500 Hz while a better servo design with a servo bandwidth of 1600 Hz can attenuate upto about 1100 Hz. It is also important to note that the sensitivity function amplifies disk vibration in certain frequency ranges. In this paper the displacement spectra for aluminum and alternate substrate materials are weighted by servo sensitivity functions of varying servo bandwidths and the closed loop TMR is computed for all the cases.

V. Results and Discussions

The measurement methods discussed above were used to compare the performance of aluminum and other alternate substrate materials. Fig. 3 shows the velocity spectrum measured

TABLE III
COMPARISON OF VARIOUS SUBSTRATE MATERIALS: HIGHER THE DISPLACEMENT METRIC, LOWER THE TRACK MISREGISTRATION

	Aluminum	Glass	Glass ceramic	Zirconia	Alumina	Silicon Carbide
Young's Modulus(Gpa)	71	100	123	223	400	430
Density d (kg/m^3)	2860	2760	2700	6040	3900	3210
Specific Stiffness (MPa m^3/kg	24.8	36.2	45.6	36.9	102.6	134.0
Poisson's Ratio	0.30	0.23	0.24	0.25	0.25	0.18
Damping Ratio(Q-factor)	0.0101	0.0092	0.0114	0.0064	0.0060	0.0035
Displacement Metric	0.79	0.97	1.49	1.52	2.56	1.56
Cumulative TMR metric	6.29	4.48	4.09	3.45	2.31	2.53

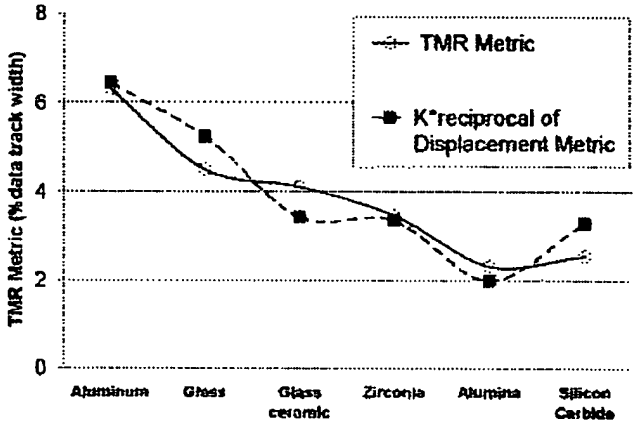


Fig. 5. The reciprocal of the displacement metric is a good predictor of TMR due to disk vibration.

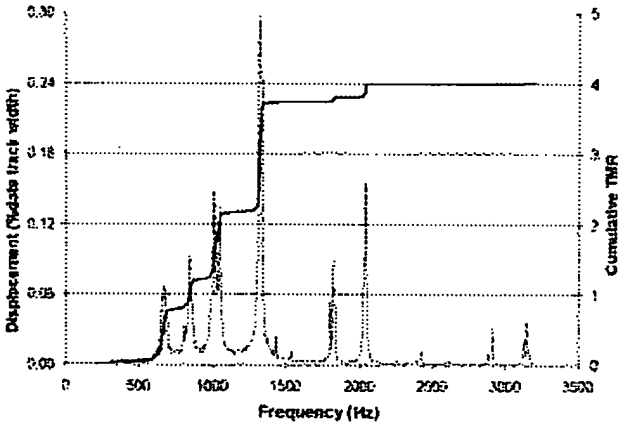


Fig. 6. The disk vibration of aluminum disks weighted by a servo-sensitivity function of 1600 Hz bandwidth and the resulting cumulative closed-loop TMR.

by the LDV for an 84 mm outer diameter (OD) and 0.8 mm thick aluminum disk.

The resonance modes have been identified on the spectrum: the umbrella mode (0,0) and the split harmonics, due to a spinning disk [2].

A. Displacement Metric as a Prediction Tool for Open Loop TMR

The total area under the power spectral density (m^2/Hz) was determined to compute the variance σ^2 and the peak cumulative 3σ TMR. Fig. 4 shows the cumulative TMR of 6.2% data track width (for 25 kTPI) for the aluminum substrate tested at 10 000 rpm.

Similarly the disk vibration spectra (refer to the Appendix, Appendix) and cumulative TMR metric for the alternate substrate materials were determined (Table III).

The Q-factor corresponding to the fundamental mode (0,0) of vibration was used to estimate the damping ratio for each substrate material [5]. Table III lists the damping ratio determined in this manner for the substrates. The highest damping ratio was for glass ceramic substrate. The displacement metric was computed [from (5)] for all the substrates. Table III also includes the specific stiffness of the substrates. Table III clearly indicates that the displacement metric is lower for substrates with higher TMR. However specific stiffness is not a clear predictor of TMR. The specific stiffness of Zirconia is lower than that of glass ceramic yet its TMR is lower. The displacement metric, which is not dependent on the density [as in (5)], is able to predict this trend better.

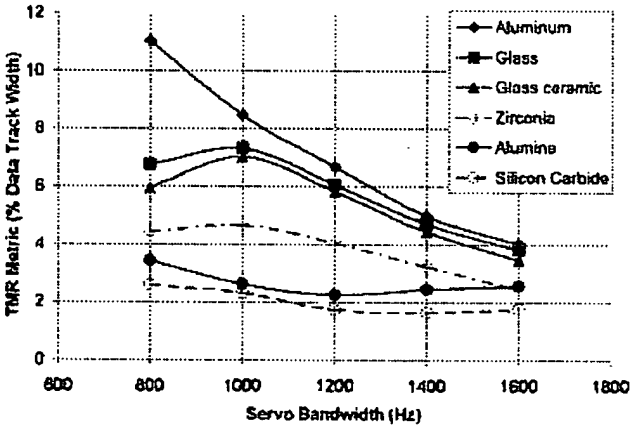


Fig. 7. The closed loop TMR metric results for the alternate substrates as a function of servo bandwidth.

Fig. 5 helps visualize the predictive nature of displacement metric. The reciprocal of the displacement metric multiplied by a constant is a good predictor of the cumulative TMR metric for all the substrates tested.

This paper identifies two approaches to reduce disk vibration based on a theoretical model. For disks of the same dimensions, the displacement metric is a good predictor of the TMR due to disk vibration for alternate disk substrate materials.

B. Prediction of Closed Loop TMR

The displacement spectra for aluminum and alternate substrate materials were weighted by servo sensitivity functions of

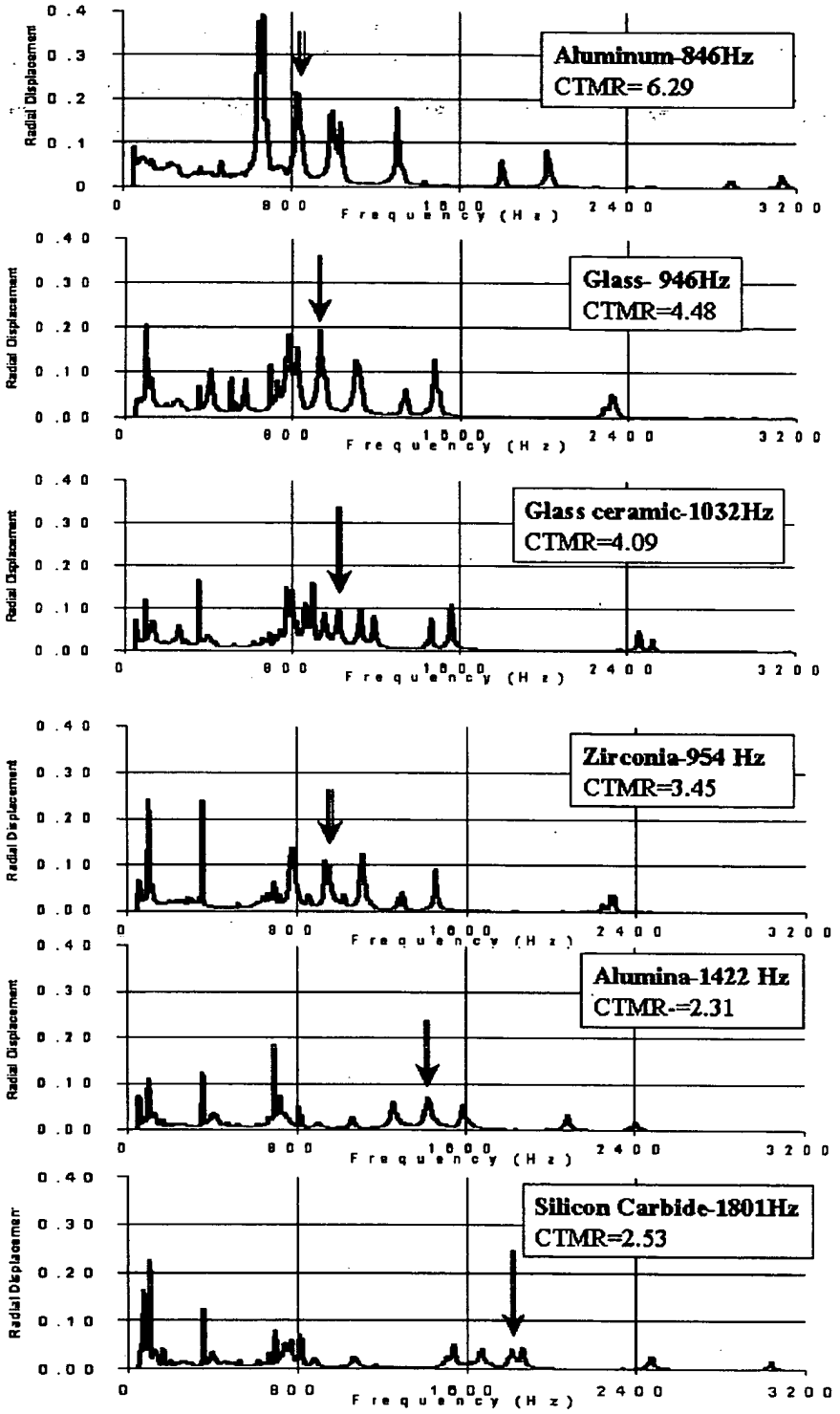


Fig. 8. Comparison of the displacement spectra of aluminum, glass, glass ceramic, zirconia, alumina and silicon carbide disk substrate materials.

varying servo bandwidths and the closed loop TMR was computed for all the cases. The low servo bandwidth (800 Hz) sensitivity function attenuates disturbances like disk vibration below 500 Hz while a servo design with a servo bandwidth of 1600 Hz can attenuate upto about 1100 Hz. It is also important to note that the sensitivity function also amplifies the disk vibration in certain frequency ranges. The displacement spectra for aluminum substrate weighted by a servo sensitivity function with a bandwidth of 1600 Hz and the closed-loop TMR metric is shown in Fig. 6. The weighted spectra shows reduced vibration amplitudes for resonant modes below 1100 Hz but amplified vibration above 1100 Hz. The closed loop TMR metric of 4.0% data

track is different from the open-loop TMR of 6.3% data track, demonstrating that it is important to understand closed-loop performance.

For a low servo bandwidth of 800 Hz, there is a large difference between aluminum and the alternate substrates, and as the servo bandwidth increases to 1600 Hz this difference is less significant (Fig. 7).

VI. Conclusion

In this paper we identified a quantitative measure called "displacement metric" that predicted closely the TMR due to disk vibration in a 10 000 rpm disk drive for aluminum, glass, glass ceramic, zirconia, alumina and silicon carbide disk substrates. The displacement metric depends on substrate material properties of Young's Modulus, damping and Poisson's ratio and is a better predictor of TMR than traditional measures like specific stiffness and is currently being used by substrate vendors to improve the performance of new alternate substrate materials. Laser Doppler Vibrometer (LDV) measurements in a 10 000-rpm disk showed that the cumulative 3-sigma TMR metric due to disk vibration was proportional to the reciprocal of the "displacement metric." The TMR metric without servo-sensitivity was found to be least for Alumina.

Closed loop TMR metric due to disk vibration was computed and compared for a range of servo bandwidths (800–1600 Hz). For a low servo bandwidth of 800 Hz, there is a large difference between aluminum and the alternate substrates, and as the servo bandwidth increases to 1600 Hz this difference becomes less. The results indicate that the servo compensator design also determines the alternate substrate that may be needed for a drive program for a required operating TMR. While previous papers [2], [3] have not addressed this issue, this is increasingly important as disk vibration is one of the largest contributors to TMR.

This work marks an important step in the development of new disk substrate materials and predicting TMR due to disk vibration for higher performance disk drives.

Appendix

Comparison of the displacement spectra of aluminum, glass, glass ceramic, zirconia, alumina and silicon carbide disk substrate materials.

Acknowledgment

The authors wish to thank J. Hawk of Quantum's Technology and Engineering Department.

References

[1] R. D. Blevins, *Formulas for Natural Frequency and Mode Shape*. Malabar, FL: Krieger Publishers, 1993.

[2] J. S. McAllister, "The effect of disk platter resonances on track misregistration in 3.5 inch disk drives," *IEEE Transactions on Magnetics*, vol. 32, no. 1, pp. 1762–1766, May 1996.

[3] J. S. McAllister, "Characterization of disk vibration on aluminum and alternate substrates," *IEEE Transactions on Magnetics*, vol. 33, no. 1, pp. 968–973, Jan. 1997.

[4] S. S. Rao, *Mechanical Vibrations*. New York: Addison-Wesley Publishing Company, 1990, pp. 83–96.

[5] P. Srikrishna and K. Kasetty, "Evaluation of damping of alternate substrate materials," in *ASME 10th Annual International Storage and Processing Systems (ISPS) Conference*, Santa Clara, June 1999.

Analytical Performance Modeling of Hierarchical Mass Storage Systems

Odysseas I. Pentakalos, *Member, IEEE*, Daniel A. Menascé, *Member, IEEE*, Milton Halem, *Member, IEEE*, and Yelena Yesha, *Senior Member, IEEE*

Abstract—Mass storage systems are finding greater use in scientific computing research environments for retrieving and archiving the large volumes of data generated and manipulated by scientific computations. This paper presents a queuing network model that can be used to carry out capacity planning studies of hierarchical mass storage systems. Measurements taken on a Unitree mass storage system and a detailed workload characterization provided the workload intensity and resource demand parameters for the various types of read and write requests. The performance model developed here is based on approximations to multiclass Mean Value Analysis of queuing networks. The approximations were validated through the use of discrete event simulation and the complete model was validated and calibrated through measurements. The resulting model was used to analyze three different scenarios: effect of workload intensity increase, use of file compression at the server and client, and use of file abstractions.

Index Terms—Mass storage systems, queuing network modeling, mean-value analysis, Unitree central file manager, compression, file abstraction.

1 Introduction

Mass storage systems are finding greater use in scientific computing research environments for retrieving and archiving data generated by model simulations in volumes on the order of terabytes. This demand on the mass storage systems is increasing at rates faster than the currently operating mass storage systems can efficiently handle [1], [2], [3]. To meet these demands, some computing centers are procuring additional storage devices without access to the tools for predicting the performance of the expected workloads on existing and new mass storage system configurations. Performance models are necessary to carry out adequate capacity planning studies for mass storage systems. Queuing network models are a viable alternative if accurate approximations can be found to deal with the features of mass storage systems which cannot be dealt with by exact models. Queuing network models can be used to provide average file storage and retrieval times and system throughput as a function of various parameters including file sizes, workload intensity, performance characteristics of the various physical storage devices that compose the mass storage system, and the architecture of the mass storage system hierarchy.

This paper describes the development of a queuing network (QN) model to assess the performance of a hierarchical mass storage system. The model was validated on the Unitree Central File Manager, used at NASA's Center for Computational Sciences (NCCS). The system being modeled consists of a large number of workstations connected to a single storage server via an Ethernet network and a Cray supercomputer connected to that same storage server via a high speed Ultranet network. The storage server is a UNIX based multiprocessor that manages the devices which comprise the hierarchy of the mass storage system. The Unitree Central File Manager (UCFM) is an application which runs on top of UNIX and manages the file systems at each level of the hierarchy, as well as the flow of files from one level of the hierarchy to another. The granularity of access to the data by Unitree is at the file level. At the particular installation where the modeling was performed, users access the mass storage system through the ftp protocol connecting to the central server at a specific port. Thus, requests for storing and retrieving files arrive in the form of *put* and *get* commands, respectively. Even though the example used in this paper is based on the UCFM, the techniques presented here can be used to model other mass storage systems that adhere to the IEEE Mass Storage System Reference Model [4].

Mass storage systems of this magnitude of storage capacity have not been available until recently, so little other work has been done to evaluate their performance. Ramakrishnan and Emer [5] developed a closed queuing network model to evaluate service alternatives for distributed mass storage systems. The performance of providing mass storage service at the level of a block versus a logical file is compared using the queuing network model. The definition of a mass storage system in [5] differs from the one considered here, since they refer to a disk based storage system without the additional complexity of traffic moving between the various layers. Drakopoulos and Merges [6],

- *O.I. Pentakalos and Y. Yesha are with the University of Maryland Baltimore County and the Center of Excellence in Space Data and Information Science (CESDIS), Goddard Space Flight Center, Greenbelt, MD 20771. E-mail: {odysseas, yeyesha}@cs.umbc.edu.*
- *D.A. Menascé is with the Department of Computer Science, George Mason University, Fairfax, VA 22030-4444. E-mail: menasce@cs.gmu.edu.*
- *M. Halem is with the Space, Data, and Computing Division, Goddard Space Flight Center, Greenbelt, MD 20771. E-mail: mhalem@gsfc.nasa.gov.*

Manuscript received 24 July 1995; revised 20 Feb. 1996.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 105197.

[7] use a closed queuing network model to evaluate the performance of a hierarchical mass storage system for various file movement criteria, and to study the trade-offs between recalling a file and accessing it by a distributed file system on various network architectures. In [6], [7] the authors focus on the network interface of the mass storage system and not on the internal interaction between the layers. We develop here an approximate closed queuing network model of Unitree. Our focus is on the performance evaluation of the interactive traffic arriving over the network in the form of ftp get and put requests. In an earlier study [8], the authors developed a trace-driven simulation of the NCCS site and concluded that the disk cache hit ratio is very small (30 percent-40 percent) and that users tend to reference either files that were just created or those which were created a long time ago (over three months). In fact, the simulation showed that, even if the disk cache were large enough to hold all references in the previous three months, the hit ratio would still remain within the 30 percent-40 percent range. Therefore, we are not including in the model the effect of file migration policies.

The rest of this paper is organized as follows. Section 2 describes in more detail the Unitree Central File Manager and its underlying hardware and software characteristics. Section 3 describes the workload imposed on the system which was used in our experiments. Section 4 describes the analytic model used. Section 5 discusses the validation of the model and several numerical results. Finally, Section 6 provides some concluding remarks.

2 Overview

In this section, we describe briefly the functionality of the UCFM and present its main hardware and software characteristics included in the model.

2.1 The Unitree Mass Storage System

The UCFM is a hierarchical distributed file system which runs as an application on top of UNIX. UCFM is a mass storage manager which provides a transparent uniform UNIX-like file system to the user. The first layer of the hierarchy consists of a pool of 75 striped magnetic disks with a total capacity of 155 GB, which behaves as a file cache for the overall system. Four robotic tape storage silos, with a capacity of 4.8 TB each, comprise the second layer. The third layer is comprised of free-standing tape storage.

The data stored on UCFM can be accessed from any local machine using either the FTP protocol or the NFS protocol. For performance reasons, only the FTP protocol method is used at NCCS. When files are first transferred to UCFM they are stored on the first layer of the hierarchy. Then, through a process called migration, a copy of the file is made available to a lower layer of the hierarchy. Based on certain configurable parameters, files from the highest layer are removed if they have not been accessed for a certain period of time. When the user tries to recall a file, UCFM retrieves the file from the highest layer on which it is located.

UCFM is composed of a number of servers running on the CONVEX machine that manages the storage hierarchy. Following the IEEE MSSRM, each server is responsible for one specific task. This distribution of responsibility and the functional separation of the components allows for load distribution, enhances the scalability of the storage system, and provides for more fault tolerance. Fig. 1 shows a diagram of the UCFM servers and their interrelation.

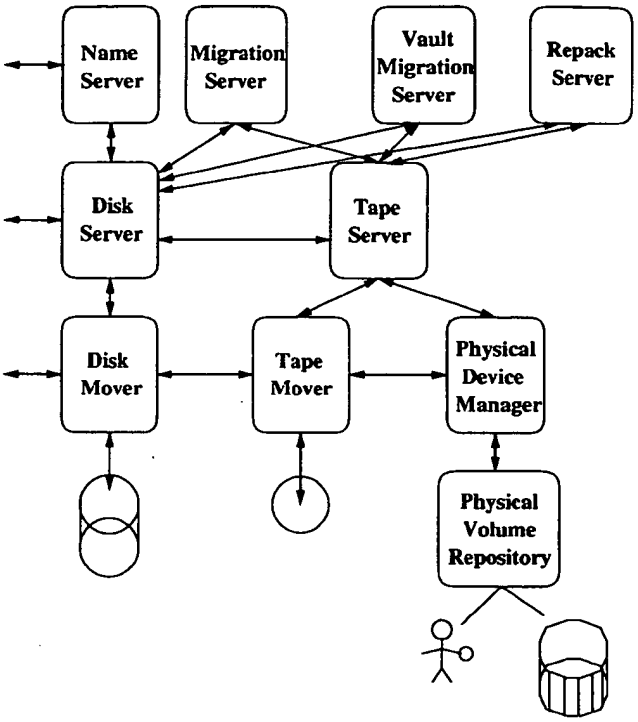


Fig. 1. UCFM system architecture.

A brief description of each of the servers in the figure follows.

Name Server: Maintains the Unitree file system structure and provides a transparent, UNIX-like interface to the Mass Storage System. It resolves human-oriented names to globally unique machine-oriented resource identifiers (bitfile ids). The Name Server maintains an on-disk database of name to bitfile id mappings, as well as an in-memory cache of recently resolved mappings. The Name Server also authenticates access rights of the requester.

Disk Server: Provides the logical means for storing and retrieving data from the disk cache. It maintains the information necessary for mapping a bitfile id into the actual file stored on the disk.

Disk Mover: Manages the transfer of file data to and from the disk cache. All such requests originate from the Disk Server. A response to each request is sent directly to the recipient of the file rather than to the Disk Server.

Tape Server: Performs the equivalent service to tapes as the Disk Server performs to the disk cache. Its objective is to maximize the use of the storage media by archiving files. It maintains all information needed to retrieve files from tapes. It receives requests from the Disk Server and the Migration Server for access to files.

Tape Mover: Manages the transfer of file data to and from tapes. It receives all its requests from the Tape Server.

Physical Device Manager: Manages the tape mounts and performs the mapping between bitfile ids and tape ids. It receives requests to mount tapes from the Tape Server and communicates its requests to the Physical Volume Repository to mount and dismount tapes.

Physical Volume Repository: Maintains the information about the location of each tape and every storage device available at the tape level. It receives requests to mount tapes from the Physical Device Manager and issues mount commands to the robot or operator.

Migration Server: Moves data from the disk cache to lower levels of storage in the hierarchy in order to increase the amount of free space on the disk cache.

Repacking and Vaulting Server: Removes the fragmentation from the tapes in the silos and performs the migration of files from the silos to the off-line tape drives.

UCFM performs four major tasks to manage the overall mass storage system. The first task is the processing of user requests for file storage and retrieval. Files which are stored into the Unitree system are always placed on the disk cache first. When the file to be retrieved is located on the disk cache, the Disk Mover is used to transmit the file to the user. If the file is not in the disk cache, it has to be brought into the disk cache from either online or off-line tape and then transmitted to the user. The second task is the migration of files from the disk cache to the silos, using various criteria for selecting the files to migrate. The criteria are that the files must reside on the disk cache for a certain amount of time before they can migrate, and there should be a certain number of files ready to be migrated before migration starts. The third task is tape repacking, which is used for removing the fragmentation from tapes and for increasing the number of free tapes at each silo. When an updated file is brought into the disk cache, the old copy of the file which possibly resided in tapes is invalidated, causing fragmentation in the tapes. Repacking starts when the number of free tapes at a silo falls below a configurable parameter and tapes which have a certain percentage of fragmentation are selected for repacking. When a tape is repacked, the data is read from the tape into the disk cache, and then stored in an empty tape. The last major task is file vaulting, which is the migration of files from robotic tapes to off-line tapes. Vaulting is performed periodically and also when repacking cannot produce the desired number of free tapes. The files are selected for vaulting using a configurable file aging parameter.

2.2 Main Hardware and Software Characteristics

The large number of devices attached to the system and their shared use by several software components creates contention at various points of the hardware architecture. To ensure that the model captured that contention, it became necessary to analyze various components of the hardware architecture at a low level.

The UCFM software runs as an application on a Convex C3830 supercomputer with three independent CPUs. Using a dynamic scheduling scheme, called automatic self-allocating processors, CPUs are assigned to processes and threads using both hardware and software interaction. Each CPU is responsible for scheduling itself to an executing process. Each thread of control posts the need for a CPU, and an idle CPU selects the thread and executes it.

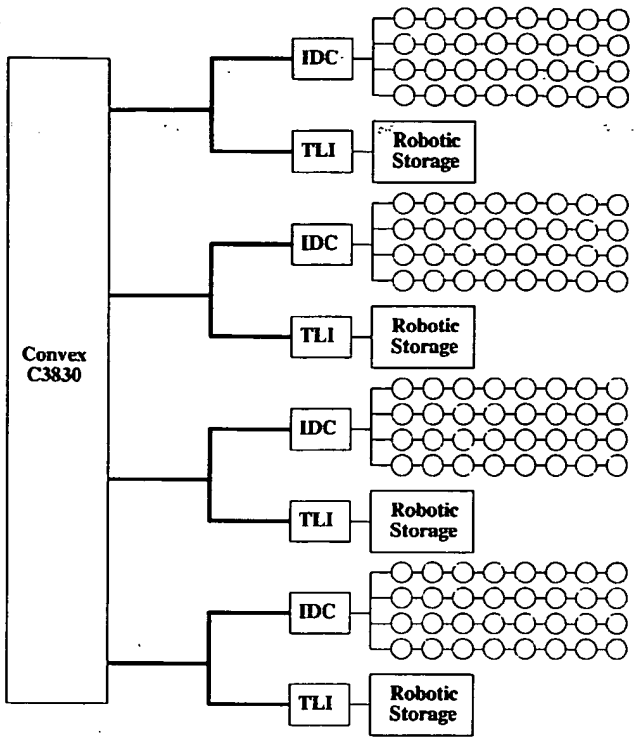


Fig. 2. I/O device connectivity.

TABLE 1
DISK CHARACTERISTICS

Disk Type	Transfer Rate (MB/s)	Average Seek (ms)	Average Latency (ms)
1	6.00	16.0	8.33
2	4.67	12.0	6.87
3	9.34	12.0	6.87
4	9.34	11.5	5.55

Fig. 2 shows the interconnection of controllers and storage devices with the I/O bus of the system. There are 128 disks attached to the system, 75 of which are used by the disk cache as striped devices. Striping will be described in a later section. As shown in Fig. 2, there are four Integrated Disk Channels (IDC), plugged into the main I/O bus of the server, which control the disks. The IDC consists of a Motorola 88100 RISC processor, separate code and data RAM, and four independent peripheral interface ports. Each peripheral interface port conforms to the ANSI Intelligent Peripheral Interface (IPI) physical interface specification and is capable of transferring data at the rate of 10 megabytes per second, concurrently with the other ports on the same IDC. Following the IPI standard, each IDC port has eight disks attached in a daisy chain fashion. Disks closer to the IDC port have higher priority over disks farther away in the chain. To ensure that this implicit priority assignment does not hinder performance, the faster disks were placed closer

to the IDC port and slower disks were placed farther away. Table 1 describes the characteristics of the four types of disks attached to the interface.

The disk cache consists of 75 disks, distributed among 15 logical striped devices of five physical disks per striped device. The five disks of each striped device were selected to be on different controller ports, thus providing the maximum possible concurrency to operations on a striped device. Striping is implemented by device drivers within the kernel and uses a Block Interleaved Distributed Parity (Raid-5) disk array [9]. Under this method, one file system block is interleaved across multiple disks. The fraction of the block size determined by the interleaving is referred to as the stripe unit. In this particular case, one block is stored across four disks. Redundancy is provided by computing the parity for each of the four stripe units, each of which resides on a different disk, and storing the result on the fifth disk of the striped device. The parity block is also distributed on different disks to provide maximum concurrency on read requests. This allows all five disks to participate on read requests. The disadvantage of RAID-5 stripping is the read-modify-write operations that occur in small writes. A write operation of one stripe unit in size requires five reads to get the information so that the new parity block can be computed, computation of the new parity, and two write operations to update the data block that was modified and to store the new parity result.

The second and third levels of the storage hierarchy consist of four robotic tape storage libraries (silos) and four free-standing tape drives. The silos manage the storage of 6,000 tapes each, providing fast robotically controlled mounting. In this specific configuration of the silos, there are six tape drives available for servicing read and write requests. All tape drives, both the ones within the silos and the free-standing ones, have the same physical characteristics. A SUN workstation is connected to both the CONVEX and the silos and maintains, for each tape, its location (robotic silo, off-line tape drive, off-line storage) and its status (mounted, stored). Requests for mounting tapes and positioning them for a read or write operation are sent to the silo by the SUN workstation. The tape drives are connected to the CONVEX through the Tape Library Interfaces (TLI) as shown in Fig. 2. Each TLI provides two independent data paths to the six tape drives inside each silo.

3 Workload Characterization

The data source for workload characterization was the log of ftp *get* and *put* requests. We were interested in evaluating the performance of the Unitree during periods of time when the interactive load was heavy. To determine the periods of heavy Unitree usage, a histogram of the *get* and *put* requests was generated for each day for a period of 10 days. Fig. 3 shows two typical histograms of interactive requests, one for *get* and one for *put* requests. Consistently, through all histograms, the interactive load on the system achieved its peak between 9:00 a.m. and 6:00 p.m. for all histograms considered. This focused our workload characterization to the requests arriving during that period of time.

The file size of both types of requests varied from small

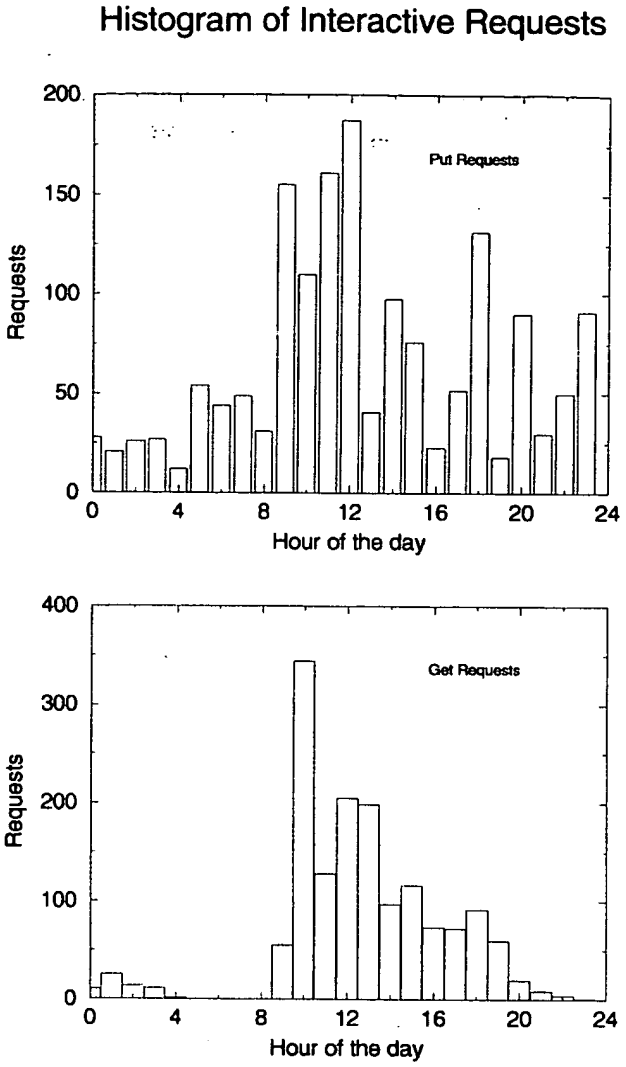


Fig. 3. Histogram of interactive requests.

files of a few kilobytes to huge files of a few hundred megabytes. Ignoring this variation in file size would introduce an error in the model, thus, the requests were separated into multiple classes with different file sizes. To determine the appropriate number of classes for each type of request and the file size for each class, a clustering analysis on measurement data was performed. We used the k-means algorithm for various values of k [10], [11]. The algorithm was initialized using k values uniformly distributed over the range of file sizes and iterated until there was no more shifting of points between clusters. The data for 10 days of get and put requests was used for the characterization. Separate data sets were generated for each day and for each of the two types of requests. The algorithm converged after five repetitions, on the average, for each data set used in our experiments. The cluster centroids provide the file sizes which determine the workload classes and the membership of points within a cluster determine the fraction of all requests for a type of that specific size.

As the number of clusters k increases, a better clustering of the points can be determined. On the other hand, as the

number of clusters increases, the computation time for solving the queuing network also increases, since the number of clusters determines the number of classes in the queuing network. In order to find a value of k which both attains a good clustering of the points and, at the same time, allows for efficient solution of the queuing network, a tightness measure was used [11]. Smaller values for the tightness imply a better clustering of the data, since the centroids are chosen to be closer to all the points within their corresponding cluster. Even though, in the limit as k approaches N, the tightness goes to zero, the decrease in the tightness is not monotonic. A locally minimal value of the tightness was observed for $k = 4$ for both get and put requests. Table 2 describes the workload selected to drive the model, showing the file size of each class selected and the frequency of occurrence of each class out of a total of 3,691 requests during the measurement period. The notation used to refer to the classes for the rest of the paper is g_1 through g_4 for the four get request classes and p_1 through p_4 for the four put request classes.

TABLE 2
WORKLOAD CHARACTERISTICS

Class	File Size in MB	Frequency of Occurrence
g_1	1.2	33.8%
g_2	19.6	9.9%
g_3	78.9	4.2%
g_4	220.6	1.4%
p_1	1.7	42.3%
p_2	34.8	3.3%
p_3	77.7	3.9%
p_4	144.1	1.2%

4 THE MODEL

4.1 Request Sequence

The objective of this study was to model the flow of the interactive requests through the various devices in the system. The fraction of time spent at each device of the system by each class determines the load demand of that class for that specific device. Some details about the functionality of the UCFM are needed here. The Name Server maintains a database of name to resource id mappings into two separate disks, both of which are not used by the disk cache and are not striped devices. Also, it maintains an in-memory cache of recently resolved mappings. One of the two disks serves as the primary database and the other serves as the secondary database, for fault tolerance reasons. The disk server maintains the necessary information for retrieving a file from the disk cache in memory. Also, all the headers of files stored in tape storage are stored on a few disks, which are, again, not used by the disk cache and are not striped devices.

The flow of both get and put requests through the system is shown below.

Get request processing sequence:

1) Use of the CPU by the ftp daemon to establish the connection.
2) Let p_{ns} be the probability that name resolution will be done by the name server's cache. With probability p_{ns}, use the CPU only to do name resolution. With probability $(1 - p_{ns})$, use the name server's disk partition to resolve the name.
3) Use of the CPU by the disk server to search the search table for the file.
4) With probability p_h, the file resides in the disk cache. So, use the striped device to retrieve the file from the disk cache.
5) With probability $(1 - p_h)$, the file is stored on tape. Let p_{ts} be the probability that the file's header information is stored in the in-memory tape header cache. With probability p_{ts}, use the CPU to obtain the file's location in tape storage.
6) With probability $(1 - p_{ts})$, use the tape header partitions on the disk to locate the file's location in tape storage.
7) With probability p_{rob}, the file is retrieved from robotic tape storage into the disk cache.
8) With probability $(1 - p_{rob})$, the file is retrieved from off-line tape storage into the disk cache.
9) Use of the CPU to transfer the file from the disk cache to the user over the network.

Put request processing sequence:

1) Use of the CPU by the ftp daemon to establish the connection.
2) As with get requests, with probability p_{ns}, use the CPU to resolve the name from the in-memory cache. Else, with probability $(1 - p_{ns})$, resolve the name from the name server's disk partition.
3) Use of the CPU to update the disk server's header information for the new file created or for the file to be updated.
4) Use of the CPU and disk to transfer the information to the disk cache.

4.2 The Queuing Network Model

Fig. 4 is a diagram of the queuing network model. Each of the major components of the model are enclosed within dotted boxes. Starting from the left, the circle labeled "User WS" is a delay server, which represents the time interval between read or write requests arriving from the user workstations. The next component is the CPU unit, which is represented by a single queue and three servers. As described in Section 2.2, the three CPUs are capable of symmetric multiprocessing, so they can be accurately modeled as three independent servers. The "Disk Cache" component represents all the striped disks that form the Unitree's disk cache. The "Disks" component represents the two disks used for storing the name server database, and the five disks used for storing the tape server search table and free space map. These disks are modeled differently from the disk cache's disks, since they are not striped devices. Finally, the "Tape Devices" component represents the robotic silo tape drives and the off-line tape drives. The next few paragraphs describe the "Disk Component" and "Tape Devices" component in more detail.

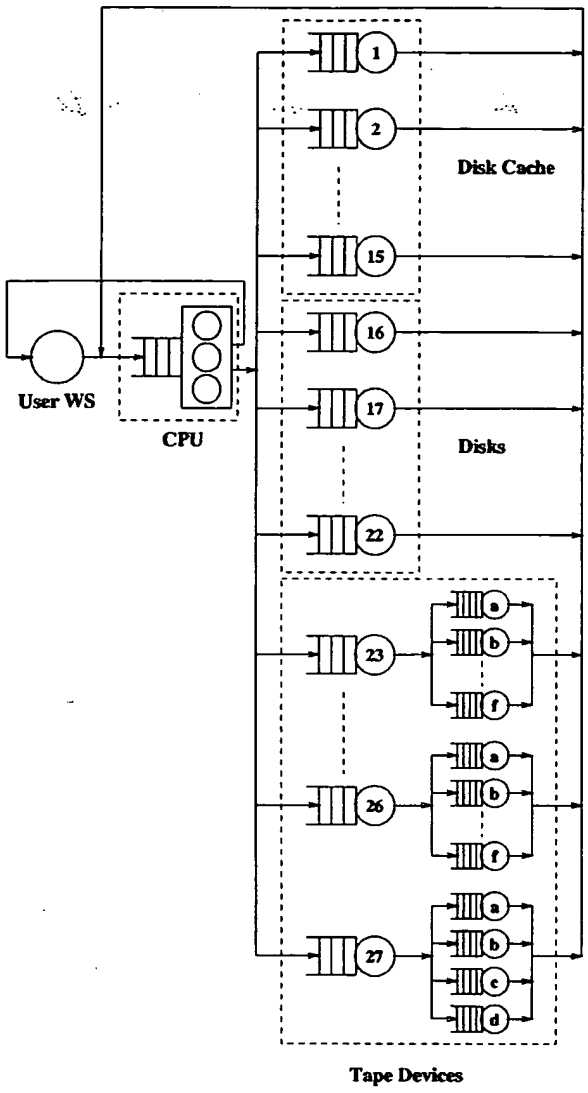


Fig. 4. Queuing network model diagram.

As described in Section 2.2, each disk is connected to the CONVEX through one of the four ports on an IDC controller, each port having eight disks connected to it. Since each of the four ports can operate concurrently with one another, the contention is at the individual port and not at the IDC controller itself. Thus, we model each disk with a single queue, since the only queuing delay is at the individual disk for servicing the requests. Using the IPI interface specification, for each port, a request is sent through the bus to one of the eight disks. The bus is then released until the disk is ready to transfer the data. Only when the disk has completed its seek and latency does it request the bus so that it can transfer the information. The bus is allocated to one of the disks, based on the priority assignment described in Section 2.2. A process-based simulation using CSim [12] was developed to model the exact operation of an IPI port to determine if modifications to the standard Mean Value Analysis (MVA) equations were necessary to model the priority. A system with a CPU, a user workstation, and a bus with eight disks was simulated. The performance parameters of the bus and of the disks used in the simulation were the same as those of the actual hardware at the NCCS site. The results of the simulation were compared with the results of the standard MVA equations, as shown in Table 3. The table shows residence time R_{disks} at the disks and the total response time R_{total} for both the analytic and simulation models for various values of the degree of multiprogramming. The total response time for the simulation also includes 95 percent confidence intervals. The last column indicates the percent error between the residence time at the disks values obtained with the two models. As can be seen from the table, the error is very small, so it was not necessary to account for bus priorities in modeling the IPI disks.

TABLE 3
IPI PORT PRIORITY MODELING

MPG	Simulation		Analytic		% Error
	R_{disk}	R_{total}	R_{disk}	R_{total}	
1	16.36	21.35 ± 0.45	16.00	21.20	2.20%
2	17.27	22.34 ± 0.59	17.54	22.79	1.56%
3	19.03	24.05 ± 0.43	19.13	24.43	0.52%
4	20.72	25.78 ± 0.50	20.77	26.11	0.24%
5	22.45	27.69 ± 0.76	22.25	27.83	1.60%
6	24.41	29.55 ± 0.61	24.17	29.58	0.98%
7	26.28	31.52 ± 0.73	25.92	31.36	1.37%
8	27.82	33.10 ± 0.82	27.69	33.16	0.47%
9	29.46	34.74 ± 0.89	29.48	34.99	0.07%
10	32.35	37.62 ± 0.84	31.29	36.83	3.28%

The 75 disks which comprise the disk cache are striped devices using RAID level 5, as described in Section 2.2. Each one of the service centers shown within the "Disk Cache" component represents the five disks which form each striped device. There has been a lot of recent work on the accurate modeling of the queuing and fork-join synchronization present in RAID level 5 disk arrays [9], [13], [14]. In order to accurately model the disk arrays in this paper, the standard MVA equations to compute the response time of each queue in the "Disk Cache" component had to be modified. The modifications made to the MVA equations and the validation of our approach are described in detail in Section 4.3.

The "Tape Devices" component represents both the robotic tape drives and the off-line tape drives, since both are connected to the CONVEX I/O bus through the TLI controllers, as described in Section 2.2. A request for a read or a write to a tape drive, regardless of whether the tape is manually or robotically mounted, requires mount time, positioning time to place the heads at the correct position of the tape, and data transfer time. When a request arrives from the Tape Server, the Physical Device Manager (PDM) issues a tape mount request. The Physical Volume Repository (PVR) determines whether the tape is located within the silo or stored on the shelf, selects a tape drive to mount the tape on, and issues the appropriate request. The queuing devices, 23-27 in Fig. 4, represent the robots that mount the tapes on the tape drives. Devices 23-26 model the robots in each of the four silos and device 27 models the human operator. Queuing devices a-f model the six tape drives

inside each of the four silos and queuing devices a-d, connected to device 27, model the four off-line tape drives.

4.2.1 Model Parameters

In this section, we describe the parameters used to solve the queuing network model and the method used for collecting them. We also define the notation used for the model's parameters used for the remainder of this paper. The CPU service times were measured using both the standard UNIX utilities and the Unitree log files [15], [16].

The "User WS" is a delay station and its service demand represents the mean interarrival time between the get or put requests, denoted by Z_r where r is the class of the job. The possible values for r can be g_1 through g_4 or p_1 through p_4. The value of Z_r was measured from the FTP log file and the interarrival time was computed using the average between arrivals of the same class for each of the classes considered. Table 4 lists the computed interarrival time for each class of requests.

TABLE 4
PER CLASS INTERARRIVAL TIMES (IN SECONDS)

	g_1	g_2	g_3	g_4	p_1	p_2	p_3	p_4
Z_r	30.0	96.1	246.7	735.7	23.3	308.5	246.7	881.0

The rest of the parameters are described using a trace through an execution scenario of a read request. When a request arrives, it requires some CPU time for interaction with the ftp daemon. This includes time for establishing the connection, authenticating the user, and parsing the request. This component of the service time is denoted as $t_{cpu,ftpo}$. After the file has been located in the storage hierarchy, it is transferred to the user via the ftp daemon, and this CPU service time is characterized by the parameters $t_{cpu,ftpta}$ and $t_{cpu,ftptb}$. File names must first be converted into resource identifiers before the system can process them. The Name Server maintains a database of file name to resource identifier records on disk and also maintains a small cache of recent requests in memory. We denote by p_{ns} the probability that the request will be resolved by the in-memory cache, and by $t_{cpu,ns}$, the service time at the CPU by the Name Server for processing the request. Also, if the request is not resolved by the in-memory cache, v_{ns} denotes the number of visits to the name server's disk partitions for resolving the request.

Using the resource identifier, the Disk Server attempts to locate the file in the disk cache and succeeds with probability p_h. In searching for the file, it consumes $t_{cpu,ds}$ seconds of CPU time. If the file is not located in the disk cache, then, with probability p_{rob}, the file is in robotic tape storage and, with probability $(1 - p_{rob})$, it is in off-line storage. The Tape Server determines the location of the file in tape storage by searching through its in-memory cache, and succeeds in finding it in the cache with probability p_{ts} or, by searching through the search table stored on the disk partitions, with probability $(1 - p_{ts})$. To retrieve the information from the disk, it visits the disks v_{ts} times. The PDM and the PVR servers then consume $t_{cpu,mount}$ CPU time for processing the mount request, which places the tape into a tape drive. The file is then transferred to the disk cache and, from there, to the user, over the network.

A summary of the input parameters to the model follows:

- Z_r: interarrival time between requests of class r.
- $t_{cpu,ftpo}$: average CPU time spent by the ftp daemon in establishing the connection and other overhead.
- $t_{cpu,ftpta}$: average CPU time spent by the ftp daemon for setting up the transfer of a file.
- $t_{cpu,ftptb}$: average CPU time spent by the ftp daemon for transferring one 64KB block of data.
- $t_{cpu,ns}$: average CPU time spent by the name server daemon for resolving the filename into a resource identifier.
- v_{ns}: number of visits to the name server's disk partitions for resolving a filename into a resource identifier.
- p_{ns}: probability that the filename to resource id mapping is in the in-memory cache rather than at one of the two disk partitions.
- p_h: probability that the file is located in the disk cache.
- $t_{cpu,ds}$: average CPU time for locating the file in the Disk Server's search table.
- p_{ts}: probability that the file's header is stored in the Tape Server's in-memory cache of file headers.
- v_{ts}: number of visits to the tape server's disk partitions for retrieving the file's header.
- p_{rob}: probability that the file is stored in robotic storage rather than off-line storage. Measurements taken by the system administrators at NCCS show that p_{rob} is equal to 0.8.
- $t_{cpu,mount}$: average CPU time spent by the PDM server and PVR servers for servicing a mount request.
- $t_{cpu,movr}$: average CPU time spent by the Tape Mover and Disk Mover in transferring one tape block from the tape to the disk cache.
- N_{silos}: the number of robotic tape silos in the mass storage system.
- N_{ontdrv}: the number of on-line tape drives in each silo.
- N_{oftdrv}: the number of free-standing off-line tape drives.
- t_{rmount}: average CPU time for robotically mounting a tape.
- t_{hmount}: average CPU time for manually mounting a tape.
- $taperate_t$: rate of transfer of data by the tape drives. For the tape drives considered here $taperate_t = 2.5MB/sec$.
- t_{tseek}: average tape seek time (equal to 1/3 of the maximum seek time [17]).
- $blocksize_d$: block size used by the disks which is 16KB.
- $blocksize_t$: block size used by the tapes which is 15KB.

4.2.2 Computation of Service Demands

This section gives the equations used to compute the service demands at each device of the queuing model using the parameters described in the previous sections. The notation for the service demands used throughout the rest of this paper is also defined in this section. The service demand at the delay center $D_{Z,r}$ for class r is equal to the time interval between arrivals Z_r for that class. The values for each class are shown in Table 4.

The service demand equation at the CPU varies depending on the request type. This happens because get and put requests follow a different path through the system during their service. For get requests, the service demand D_{cpu,g_i} at the CPU for class g_i is computed using the equation:

$$D_{cpu,g_i} = t_{cpu,ftpo} + t_{cpu,ftpta} + t_{cpu,ftptb}\left\lceil \frac{filesize_i}{blocksize_d} \right\rceil + t_{cpu,ns} + t_{cpu,ds} + (1-p_h)\left(t_{cpu,mount} + t_{cpu,movr}\left\lceil \frac{fliesize_i}{blocksize_i} \right\rceil \right). \quad (1)$$

This equation accounts for all the CPU time used to service a get request: The time spent at the ftp daemon, at the Name Server, at the Disk Server, at the PDM and PVR servers, and the time to transfer from the tape to the disk cache and from the disk cache to the client. The actual values depend on the file sizes for each of the classes, since $t_{cpu,ftpb}$ and $t_{cpu,movr}$ are service demands per block. For put requests, the service demand D_{cpu,p_i} at the CPU for class p_i is computed using the equation:

$$D_{cpu,p_i} = t_{cpu,ftpo} + t_{cpu,ftpta} + t_{cpu,ftptb}\left\lceil \frac{filesize_i}{blocksize_d} \right\rceil + t_{cpu,ns} + t_{cpu,ds}. \quad (2)$$

Since files stored into the Unitree are always placed in the disk cache, the service demand equation for put requests is simpler. The total service demand accounts for CPU time spent at the ftp daemon, at the name server, at the disk server, and for transferring the file. Again, the file transfer time depends on the file size for the specific put class.

The service demand equations for the striped devices also depend on the type of request. For both get and put requests, the service demand for each striped device is a function of the service demand of each disk comprising the striped device. The notation used here is explained in detail in Section 4.3. The service demand D^j_{i,g_i} of a get class g_i job at physical device j, which is a component of striped device i, can be computed using the following equation:

$$D^j_{i,g_i} = p_h(1/STR)\left[(seek_j + lat_j)\phi(g_i) + \frac{blocksize_d}{trate_j}\left\lceil \frac{filesize_{g_i}}{blocksize_d} \right\rceil \right], \quad (3)$$

where $seek_j$ is the average seek time, lat_j is the average latency, $trate_j$ is the transfer rate of device j, and $\phi(r)$ is a function that gives the number of seeks needed to sequentially access a file of size $filesize_r$. The Unitree file system uses a varying block size allocation method. On the first two requests for a block, a 64KB block is allocated. For every subsequent request after that, the size of the block allocated is doubled until it reaches a value of 4MB and remains fixed after that. We assume that a seek and a rotation are only needed for positioning the heads at the start of each of the varying size blocks, while access to data within the blocks is sequential. Using the size of the allocation units for the Unitree file system, $\phi(r)$ is given in Table 5. We assume that the data is evenly distributed among the STR striped devices. The service demand D^j_{i,p_i} of a put class p_i job at physical device j, which is a component of striped device i, is computed as

$$D^j_{i,p_i} = p_h(2/STR)\left[(seek_j + lat_j)\phi(p_i) + \frac{blocksize_d}{trate_j}\left\lceil \frac{filesize_{p_i}}{blocksize_d} \right\rceil \right]. \quad (4)$$

The reason why the multiplier is one for get classes and two for put classes is explained in Section 4.3. Table 1 shows the average seek, average latency, and transfer rate for each of the disk types. Once the service demand of each physical device has been computed, the service demand D_{i,g_i} on the logical striped device i by class g_i of get requests is computed using the equation:

$$D_{i,g_i} = H_5 D^*_{i,g_i}, \quad (5)$$

where $H_k = \sum_{i=1}^{k} 1/i$ and D^*_{i,g_i} is the service demand of a class g_i request on any of the physical disks which comprise striped device i. H_k is used to estimate the expected maximum of k independent exponentially distributed service times [18]. The service demand D_{i,p_i} by class p_i of put requests is computed using the equation:

$$D_{i,p_i} = H_5 D^*_{i,p_i}. \quad (6)$$

Equations (5) and (6) are explained in detail in Section 4.3, along with the notation used in the equations.

TABLE 5
AVERAGE NUMBER OF SEEKS PER CLASS

Class r	$\phi(r)$
g_1	6
g_2	11
g_3	26
g_4	60
p_1	6
p_2	15
p_3	25
p_4	42

Disks 16 and 17 in Fig. 4 are used by the Name Server for storing its database. We make the assumptions that $p_{ns} = 0$ and that, in order to resolve a filename, the name server needs to make two visits to one of the two disks. In an earlier study [8], it was found that this mass storage system is used as an archive and files are retrieved after they have resided in the mass storage system for a long period of time. Also, once they are retrieved, they are not accessed again for a long time. This prevents the Name Server's cache from reducing the number of disk accesses. After a few hours of operation, the Name Server has in-cache entries for the top level directories, but it still needs to make one disk reference for the user's home directory resource id, plus another disk reference to get the user's file resource id. The equation for computing the service demand $D_{i,r}$ of class r at device i, where $i \in \{16, 17\}$, is:

$$D_{i,r} = 2\frac{1}{2}\left(seek_j + lat_j + \frac{blocksize_d}{trate_j} \right)(1 - p_{ns}). \quad (7)$$

It is multiplied by two because two visits are required per request and by 1/2 since we assume the entry is equally likely to be in either one of the two disks.

Disks 18 and 19 are used by the Tape Server for storing its search tables and disks 20, 21, and 22 are used by the

Tape Server for storing file headers for each of the files stored on tape. Using the same reasoning as in the previous paragraph, we make the assumption that $p_{ts} = 0$. We also make the assumption that the tape server needs to make one visit to one of the two search table disk partitions and one visit to one of the three header disk partitions for each request that it services. This is a reasonable assumption, since a hashing algorithm is used in memory to find the block on disk, where the file's header information must be within the search table disk partition, and, then, once that is known, the exact block where the header is located is also known. The service demand at disks 18 through 22 is zero for the write classes $(p_1, \cdots, p_4)$ since files are written to the disk cache first. The equation for computing the service demand $D_{i,r}$ of class r $(r = g_1, \cdots, g_4)$ at device i, where $i \in \{18, 19\}$, is:

$$D_{i,r} = \frac{1}{2}\left(seek_j + lat_j + \frac{blocksize_d}{trate_j}\right)(1 - p_h)(1 - p_{ts}) \quad (8)$$

and the equation for the load demand $D_{i,r}$ for $i \in \{20, 21, 22\}$ and $r = g_1, \cdots, g_4$ is:

$$D_{i,r} = \frac{1}{3}\left(seek_j + lat_j + \frac{blocksize_d}{trate_j}\right)(1 - p_h)(1 - p_{ts}). \quad (9)$$

Again we make the assumption that the information is evenly distributed among the disks.

The TLIs are modeled using two components, as explained in Section 4.2. Devices 23 through 27 from the queuing network diagram in Fig. 4 represent the robotic and human mount servers. The service time for $i \in \{23, 24, 25, 26\}$ is given by t_{rmount} and the service time for $i = 27$ is given by t_{hmount}. Using these values for the service times, the service demands $D_{i,r}$ for class $r = g_1, \cdots, g_4$ for devices $i \in \{23, \cdots, 26\}$ are given by:

$$D_{i,r} = (1 - p_h)\frac{1}{4}p_{rob}t_{rmount} \quad (10)$$

and, for $i = 27$, by:

$$D_{i,r} = (1 - p_h)(1 - p_{rob})t_{hmount}. \quad (11)$$

These service demands are all zero for classes $p_1, \cdots, p_4$, since file puts go to the disk cache and not to the tapes. We assume that accesses to robotic tape drives are evenly spread among the four silos and the six tape drives. This is a reasonable assumption since the Unitree attempts to perform load balancing by spreading the load among all available resources. Finally, the service demand at each of the tape drives within the "Tape Devices" component is simply the amount of time it takes to seek to the position of the file's data within the mounted tape t_{tseek} plus the amount of time to transfer the file from the tape to the disk cache. Thus, the service demand $D_{i,r}$ for class $r = g_1, \cdots, g_4$ at device $i \in \{23a - f, 24a - f, 25a - f, 26a - f\}$ is

$$D_{i,r} = (1 - p_h)p_{rob}\frac{1}{N_{silos}N_{ontdrv}}\left(t_{tseek} + \frac{blocksize_t}{taperate_t}\left\lceil\frac{filesize_r}{blocksize_t}\right\rceil\right), \quad (12)$$

for $i \in \{27a, 27b, 27c, 27d\}$ is

$$D_{i,r} = (1 - p_h)(1 - p_{rob})\frac{1}{N_{ontdrv}}\left(t_{tseek} + \frac{blocksize_t}{taperate_t}\left\lceil\frac{filesize_r}{blocksize_t}\right\rceil\right), \quad (13)$$

and is zero for the write classes since file puts are done on the disk cache.

4.3 Modeling of Striped Devices

Striped devices cannot be modeled directly with the MVA equations since they exhibit fork-join synchronization, which does not satisfy product-form conditions; each request is broken up into multiple independent requests which must all complete before the original request is complete. RAID level 5 presents the additional complexity that each write request of a small amount of data, relative to the stripe unit, results in a read-modify-write cycle. In the Unitree system, the block size used for reading and writing data is 64KB and the stripe unit is 16KB. Each logical striped device consists of four disks for data and one disk for the parity block, keeping in mind that the parity block rotates among the five disks. In the case of read requests for a single 64KB block, five independent requests will be generated, and the original request will not be served until both requests have completed. In the case of write requests to a 64KB block, five independent read requests are generated to read the 64KB of data plus the parity block, the new parity is computed, and then five write requests are generated to write the 64KB of data back plus the new parity block. Fig. 5 shows an example where stripe units 2 and 3 have been modified. Five reads are generated, and, after they synchronize, the new parity is computed in the block labeled PC. Then, five stripe units are written, even though only stripe units 2, 3, and the parity P changed.

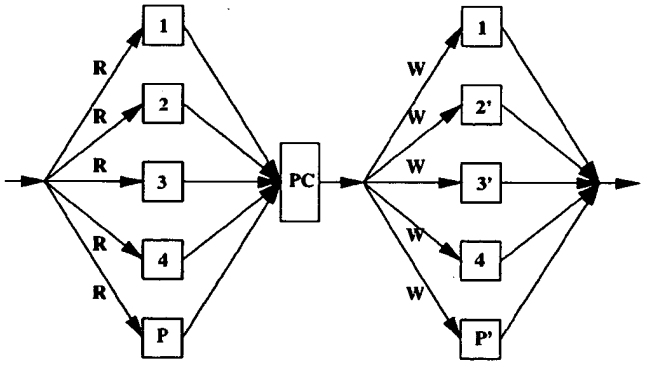


Fig. 5. Processing of a one block write by the striped device.

In order to accurately model the striped devices, we modified the equation for computing the response time in the MVA equations. We first introduce some notation.

ND: Number of disks forming the logical striped device. In our case, ND is equal to five. For each stripe unit, ND-1 disks contain data and the other contains the parity.

K: Number of physical devices involved in satisfying a file system block request ($K \leq ND$). As described above, for get requests, $K = 2$ and, for put requests, $K = 5$.

$\mathcal{D}_i$: Set of physical disks j that form logical striped device i.

STR: Number of striped logical devices. In our case, *STR* is equal to 15.

$R^*_{i,r}$: Residence time of logical striped device *i* for class *r*.

$D^*_{i,r}$: Service demand of a class *r* request on any of the physical disks which comprise striped device *i*. Since all the physical disks which form a striped device are identical and, since we make the assumption that the data is evenly distributed among the disks, $D^*_{i,r} = D^j_{i,r}\ \forall j \in \mathcal{D}_i$.

$\vec{n}^*_i(\vec{N} - \vec{1}_r)$: Average number of jobs at any of the physical devices which comprise device *i*.

Every request to the striped device generates K independent requests. The time it takes to service the original request is equal to the maximum of the residence times of the K requests. We thus set the response time of the logical striped device to:

$$R^*_{i,r}(\vec{N}) = D_{i,r}\left[1 + \vec{n}^*_i(\vec{N} - \vec{1}_r)\right].$$

The service demand, $D_{i,r}$, of the logical device is a function of the service demand $D^*_{i,r}$ of any of the physical disks which comprise the striped device. For get classes, (5) is used for computing D_{i,g_i} for logical device i. For put classes, (6) is used for computing D_{i,p_i}. In both equations, we make the assumption that the data is distributed evenly among all striped devices, which explains the use of the factor $1/STR$ in (3). In both get and put requests, five physical devices are involved in servicing one block request; thus, we used H_5. Also, for write requests, since there must be one round of five read requests followed by a round of five write requests, we multiplied the overall service demand by two.

A process-oriented simulation using CSim [12] was developed to validate this approximation. A system with a CPU, a user workstation, and a five disk array was simulated, and its output was compared with the results of the modified MVA equations as shown in Table 6. The table shows the residence time R_{raid} and the total response time R_{total} for both the analytic and simulation models for various values of the degree of multiprogramming. The total response time for the simulation also includes 95 percent confidence intervals. The last column indicates the percent error between the residence time at the RAID disk values obtained with the two models.

TABLE 6
RAID-5 MODELING

MPG	Simulation		Analytic		% Error
	R_{raid}	R_{total}	R_{raid}	R_{total}	
1	45.94	115.81 ± 1.72	45.67	115.67	0.58%
2	51.85	124.28 ± 2.24	54.13	127.83	4.39%
3	59.56	136.94 ± 3.03	63.60	141.46	6.78%
4	68.30	149.51 ± 1.99	73.92	156.29	8.22%
5	78.04	164.84 ± 1.12	85.13	172.42	9.08%
6	88.82	180.47 ± 2.26	96.77	189.16	8.87%
7	100.81	197.36 ± 1.54	108.84	206.51	7.96%
8	112.62	215.20 ± 2.66	121.24	224.34	7.65%
9	124.36	230.90 ± 2.14	133.89	242.53	7.66%
10	137.49	250.68 ± 3.78	146.75	261.02	6.73%

As it can be seen from the table, the approximation provides errors below the 10 percent level for all values of the degree of multiprogramming.

5 NUMERICAL RESULTS

This section presents the measured parameters for the model, discusses the model validation and calibration, and presents the analysis of three different scenarios. The first scenario explores the effect of workload intensity increase on file transfer time and throughput. The second discusses the advantages of using two different types of compression strategies, and the third analyzes the benefits of using file abstractions.

Table 7 shows the actual measured values for the parameters used to compute the service demands at all devices. All the parameters were collected using the standard UNIX utilities and the Unitree log files. Using the equations for service demands for each device described in Section 4.2.2, the model was parameterized and solved using the modified MVA equations. Due to the large number of devices and classes, the approximate MVA algorithm was used, which converged in only three to five iterations.

TABLE 7
MEASURED PARAMETER VALUES

Parameter	Value (in secs)
$t_{cpu,ftpo}$	0.007229
$t_{cpu,ftpta}$	0.00036
$t_{cpu,ftptb}$	0.0000618
$t_{cpu,ns}$	0.2070
$t_{cpu,ds}$	0.6776
$t_{cpu,mount}$	0.506
$t_{cpu,movr}$	0.005294
t_{rmount}	79.6
t_{hmount}	118.8

Validation and calibration is a necessary step in model development [19]. Validation and calibration require that the system be measured so that the measured performance metrics can be compared with the ones obtained by the model. The first two calibration efforts were aimed at fine tuning the service demands at the disks and tapes as a function of the file size. The hit ratio, p_h, was first set to one to eliminate the tape subsystem. It was observed that the service demand for the small file classes of get requests (g_1 and g_2) was smaller than the measured value. Therefore, the value of the function $\phi(r)$ was increased to 12 and 16, respectively, to make the service demands match. Next, the hit ratio was made equal to zero to introduce the tape subsystem and eliminate the disk cache for the get classes. This allowed us to calibrate the tape transfer rate parameter $taperate_i$. This parameter was calibrated at 1.48 MB/sec to make the transfer time computed by the model match the measured value. This calibration was later justified by independent measurements taken by the system administrators at the NCCS site. Note that the calibrated value for the tape transfer rate is smaller than the manufacturer's advertised value. Finally,

TABLE 8
MODEL VALIDATION RESULTS (TRANSFER TIMES IN SECONDS)

	$p_h = 1$			$p_h = 0$		
Class	Meas. Results	Model Results	% Diff.	Meas. Results	Model Results	% Diff.
g_1	1.42	1.42	0.00%	85.57	92.20	7.74%
g_2	6.35	5.73	9.76%	107.10	111.06	3.70%
g_3	18.04	18.04	0.00%	183.97	171.56	6.75%
g_4	48.26	48.27	0.02%	328.61	316.14	3.79%
p_1	1.78	1.83	2.81%	2.02	1.82	9.41%
p_2	10.16	11.12	9.45%	10.60	10.95	3.30%
p_3	21.60	23.21	7.45%	21.74	22.85	5.11%
p_4	44.86	42.07	6.22%	44.15	41.41	6.21%

the technique of calibration by adjusting the multiprogramming levels [19], [20] was used to calibrate the model for low values of the workload intensity. The adjustments implied an upward change of at most two in the multiprogramming level of three classes. As the workload intensity increased, such a calibration was not necessary.

A synthetic workload that mirrors the real workload, characterized in Section 3, was developed and measurements were taken for the purpose of validating the model. This synthetic workload generates a large number of requests for each of the classes. Table 8 shows a comparison between measured and computed file transfer times for all classes and for the two extreme values of the hit ratio. All the measurements were taken during times of the day when the system was otherwise idle to reduce the effect of external requests on the requests generated by our synthetic workload. The second column in Table 8 lists the average of measured values for requests from files that were stored in the disk cache. Column 5 lists the average of measured values for requests that were satisfied from the tape subsystem. Columns 3 and 6 list the values computed, with the calibrated model for hit ratios of one and zero, respectively. Finally, columns 4 and 7 show the percent difference between the measured values and the results computed with the calibrated model. The largest errors were 9.76 percent and 9.41 percent, both of which occurred in small classes.

The validated model is used in the following subsections to analyze three different scenarios.

5.1 Effects of Workload Intensity Increase

In a report by The Computer Environments and Research Requirements Committee, predictions are made on the computing requirements for 1997-2004 on the data storage and retrieval systems, for supporting NASA earth and space science research [21]. According to this report, it is expected that, in the next two years, the archival and retrieval rate on the UCFM will increase by a factor of five. In order to determine acceptable service levels for the eight classes in our workload, we conducted an e-mail survey of all users of the mass storage system at NCCS. We obtained responses from 17 percent of the 450 users who received the survey. The responses indicated the acceptable mean transfer time for each of the eight classes.

To examine the performance of the mass storage system under an increased workload intensity scenario, we varied the load factor M_r, defined as the maximum number of simultaneous class r requests present in the mass storage system. The baseline load factor mix was determined using the frequency of occurrence of each class, shown in Table 2. The values used for the multiprogramming level were $M_{g_1} = 34$, $M_{g_2} = 10$, $M_{g_3} = 4$, $M_{g_4} = 1$, $M_{p_1} = 42$, $M_{p_2} = 3$, $M_{p_3} = 4$, and $M_{p_4} = 1$. Figs. 6 and 7 show the variation of the average file transfer time versus the multiplier of the baseline load factor mix for classes g_3 and g_4 in the first figure and p_3 and p_4 in the second figure, for a hit ratio of 0.3. As can be seen in the figures, the increase in transfer time, due to a five-fold increase in the load factor, will be 19 percent for class g_3, 28 percent for class g_4, 48 percent for class p_3, and 47 percent for class p_4. The smaller get and put classes are not as affected by the increase in the load factor. The coefficient of variation in the survey responses for get classes was higher than that for put classes. Therefore, we assumed as the acceptable value for the service level the mean plus two standard deviations in the case of get classes and just the mean in the case of put classes. The threshold for the get classes was derived as a compromise between the user community desired service levels and the constraints imposed by the resources and budget of the existing installation. For class g_3, the service level of 174 seconds is violated when the load multiplier is 7.5. For class g_4, the service level of 406 seconds is violated for a load multiplier of 8.5, as can be seen from Fig. 6. For class p_3, the service level of 52 seconds is violated for a load multiplier of eight, while, for class p_4, the service level of 80 seconds is violated for a load multiplier of seven, as shown in Fig. 7.

Fig. 8 shows the transfer time of get classes as the hit ratio varies between 0.0 and 1.0, with the multiprogramming level set to that used to validate the model. The effect of the hit ratio on the transfer time depends on the service demands at the disk cache and tape subsystem for each class. As shown in Fig. 8, the transfer time decreases as the hit ratio increases. This can be better understood by looking at

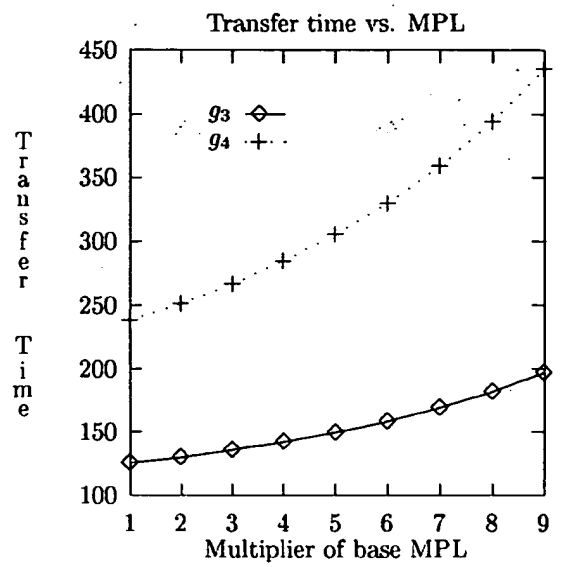


Fig. 6. Transfer time (in seconds) vs. load factor multiplier.

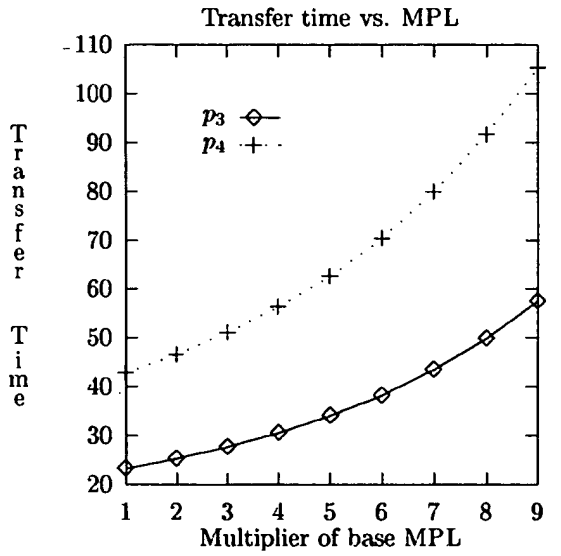


Fig. 7. Transfer time (in seconds) vs. load factor multiplier.

Fig. 9, which displays the variation of the service demand for the disk cache and the tape system for class g_4 as a function of the hit ratio p_h. Also shown in the figure, is the sum of the service demands for the disk cache and tape system. As can be seen, as the hit ratio increases, the overall service demand of the combined disk cache and tape system decreases. This explains why the transfer times decrease as a function of the hit ratio. The same kind of behavior was observed for all classes. The factor by which the transfer time decreases is larger for the small file classes. For example, for class g_1, the transfer time at hit ratio one is 66 times smaller than the same value at hit ratio zero, while, for class g_4, the reduction factor is 6.6. The explanation for this effect stems from the fact that, for the larger files (classes g_3 and g_4), the overhead of mounting a tape is amortized over a larger number of blocks than for the small file classes. Tape devices store data contiguously on tape and only exhibit a seek delay initially, when positioning the heads at the beginning of the file's data. This causes the effective transfer rates of tape drives to be comparable to those of disk devices, since disks exhibit per block seek and latency overheads. Also, from this figure, we can infer that an increase in the hit ratio from the current measured value of 30 percent to 50 percent would decrease the transfer time of a class g_3 request by 25 percent and of a class g_4 request by 23 percent. This provides incentive for research in prefetching techniques for caching.

5.2 Analysis of File Compression

We analyze here the effects of using file compression in two different manners:

1) *Client Compression:* In this scenario, files are compressed and decompressed at the client [22]. Thus, before storing a file into the mass storage system, the file is compressed at the client. Files are retrieved from the mass storage system in compressed form and decompressed at the client. The model is modified as follows, to account for this type of scheme: All service demand equations that depend on file sizes have the file size reduced by a file compression ratio f_{cr}, which now becomes a parameter of the model. The compression and decompression times at the client are not taken into account when computing the transfer time under the client compression scenario. This was done because files are usually retrieved from the mass storage system in batch mode. Once retrieved by a client, they are used many times. So, the decompression time at the client is amortized over many uses of the file. Besides, different clients would give different values for the compression and decompression time. Use of compression is mainly geared toward increasing the effective storage capacity of the mass storage system as opposed to improving its performance.
2) *Server Compression:* In this scenario, files are stored in the mass storage system in uncompressed form (in the disk cache). After the file has been unreferenced for a certain amount of time, it is compressed and remains in the disk cache (this is done during off-peak periods). The migration algorithm migrates the file in compressed form to the tapes. A get request that finds a file in the cache may find the file in compressed or uncompressed form. If the file is compressed, it has to be decompressed before it is transferred to the client. Under this scenario, compression is not relevant, since it is done during off-peak periods. Decompression, however, has to be taken into account. To obtain the parameters for the decompression time at the server, we used the UNIX compress utility [23], which is based on a variation of the Ziv-Lempel sliding window, directory-based, data compression algorithm [24], and measured the decompression times for various file sizes. Linear regression was applied to generate the following function that gives the decompression time at the server (Convex):

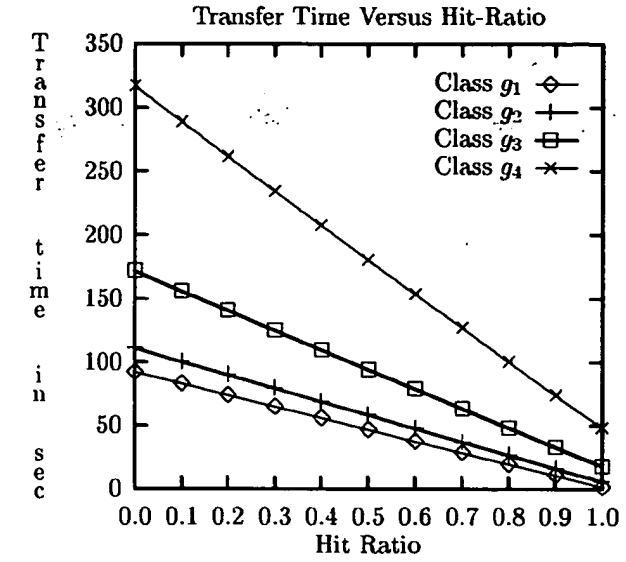


Fig. 8. Transfer time (in seconds) of get classes vs. hit ratio.

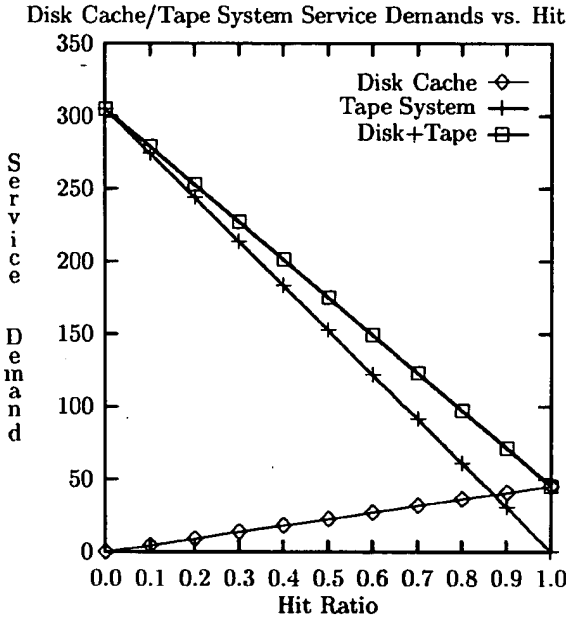


Fig. 9. Disk cache and tape system service demands vs. hit ratio.

$$D_d^s = 1.14 * \textit{filesize} - 0.058 \text{ in sec}, \quad (14)$$

where *filesize* is expressed in megabytes and the decompression time in seconds.

In this scenario, we need to add two additional parameters to the model: the compression ratio f_{cr} and the probability P_{cc} that a file is compressed given that it is found in the cache. The service demands for the get classes at the CPU (1), and physical disk devices (3) were recomputed using the following equation:

$$D_{i,g_i} = (1 - P_{cc})D_{i,g_i}^{fs} + P_{cc}D_{i,g_i}^{cfs}, \quad (15)$$

where D_{i,g_i}^{fs} is the equation with the original file sizes and D_{i,g_i}^{cfs} is the equation with the compressed file sizes. For the TLI devices (12), the equation is the same as in the original model, but uses the compressed file sizes. The service demands for the put classes are not affected in this scenario, since files are stored in the mass storage system in uncompressed form.

To assess the merit of the compression schemes, we defined a metric called gain, G_r, defined as:

$$G_r = 100 \times \left(T_r - T_r^c\right)/T_r, \quad (16)$$

where T_r is the file transfer time for class r without compression and T_r^c is the transfer time for class r with the compression scheme. G_r stands for the percent reduction in transfer time.

Fig. 10 shows the gain G_r as a function of the compression ratio f_{cr} for all four get classes under client compression. Larger classes benefit more from client compression, since a large fraction of the service demand imposed by these classes on the system depends on the file size rather than constant factors, such as mount delays. For class g_4, a realistically attainable compression ratio of 30 percent can offer a performance gain of 23 percent. On the other hand, smaller classes do not gain much from compression. Put classes exhibit similar behavior as get classes, although their gain is greater than that of get classes for the same compression ratio. The reason for this is that put classes are only affected by the disk cache whose response time is a function of the file size, whereas get classes are also affected by constant delays at the tape subsystem.

The merits of server compression are explored in Fig. 11, which shows the gain G_r for class g_4 as a function of the compression ratio f_{cr} for four different values of P_{cc} (the probability that the file is found in compressed form at the disk cache). Decompression at the server imposes a considerable load on the CPU that cannot be offset by the decreased file

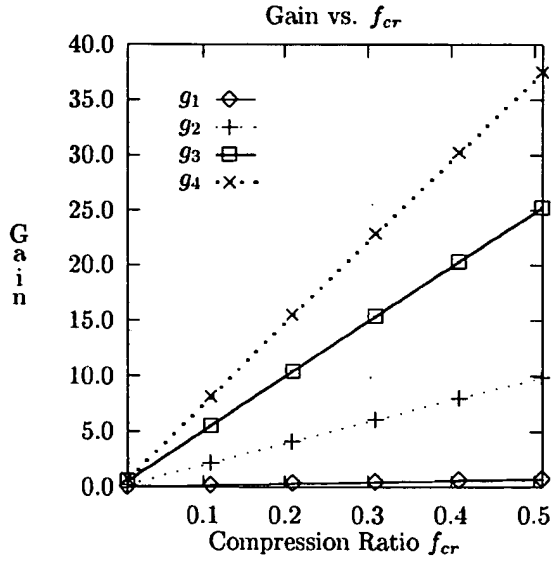


Fig. 10. Client compression gain vs. f_{cr} for all classes.

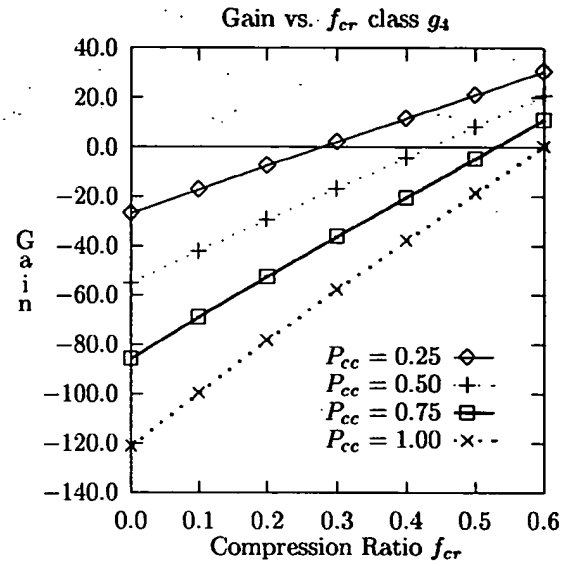


Fig. 11. Server compression gain vs. f_{cr} for class g_4.

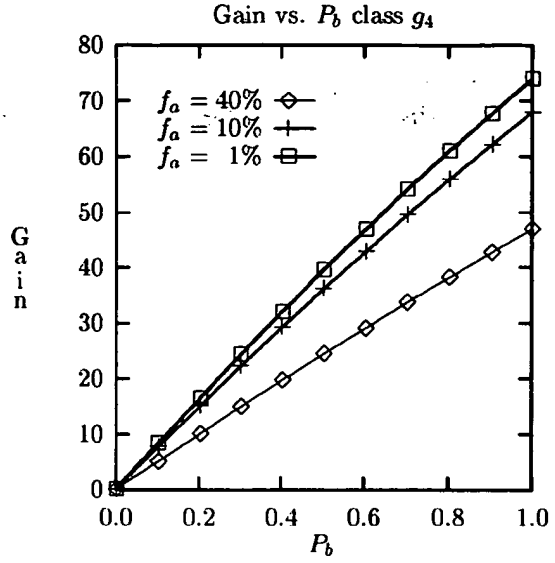


Fig. 12. Abstraction gain vs. P_b for class g_4.

size for large values of P_{cc}. This method can be beneficial only if the value of P_{cc} is kept below 25 percent for an average compression ratio of at least 30 percent. In order for the value of P_{cc} to remain below 25 percent, the amount of time that the file resides in the disk cache in compressed form before being migrated should be adjusted by appropriate system software. The other get classes also exhibit similar behavior, whereas put classes are not affected by this scenario.

5.3 Analysis of File Abstraction

Abstraction files are reduced versions of original files used for browsing purposes [1]. According to [15], most NASA image data can be significantly compressed (i.e., by a factor of 20 or more) without significant loss in the visual presentation of the data, using lossy techniques, such as wavelet encoding and vector quantization. By servicing a number of requests with abstraction files, rather than the complete copies, the load imposed on the tape devices of a mass storage system can be reduced considerably, thereby reducing the transfer time experienced by most users. In order to evaluate the effect of file abstraction on our model, four more get classes were added to represent the requests serviced by abstraction files. Two new parameters, P_b and f_a, have to be defined. P_b is the probability that an access can be satisfied by an abstract version of the object and f_a, the file abstraction factor, is the factor by which a file is abstracted. So, an $f_a = 1\%$ implies that the abstract file is one percent of the original file in size. The service demands for the original get classes remain as before. The service demands for the new abstract classes are computed using the same equations used for the original get classes, but with smaller file sizes to represent the abstracted form of the files. The load factor (M_r) for the original classes is equal to its original value multiplied by $(1 - P_b)$, while M_r for the abstraction classes is equal to the original value times P_b.

To assess the merit of the abstraction classes of requests we defined a new metric called gain, G_r, defined as

TABLE 9
ABSTRACTION GAIN FOR GET CLASSES AT f_a = 10 PERCENT

P_b	g_1	g_2	g_3	g_4
0.1	0.17	2.17	5.33	7.75
0.3	0.49	6.26	15.39	22.36
0.5	0.79	10.15	24.96	36.27
0.7	1.08	13.86	34.07	49.51
0.9	1.36	17.39	42.75	62.13

$$G_r = 100 \times \left(T_r - T_r^a\right)/T_r, \tag{17}$$

where T_r is the transfer time of the original get class r without abstraction and T_l^a, the transfer time under the abstraction scenario, which is computed as

$$T_r^a = (1 - P_b)T_r^{orig} + P_b T_r^{abs},$$

where T_r^{orig} is the transfer time of the original get class r and T_r^{abs} is the transfer time of the corresponding abstraction class. Note that both T_r^{orig} and T_r^{abs} are computed under the abstraction scenario. The gain G_r stands for the percent reduction in transfer time.

Fig. 12 shows the gain G_r for class g_4 as a function of the browse probability P_b for three values of the abstraction ratio f_a. As expected, the gain increases with the browse probability and with the compression ratio. For this class, reasonable gains can be achieved, even for not large values of the browse probability. For example, for a browse probability of 40 percent and an abstraction ratio of one percent (not unrealistic for image files), a reduction of 32 percent in transfer time can be obtained. The gain decreases for small file classes. Table 9 shows the gain for classes g_1 through g_4 and for different values of P_b. As can be seen from the table, for $f_a = 10\%$, the maximum gain for the small file class g_1 is 1.4 percent, while, for the large file class, g_4 is 62.1 percent. From these figures, it is clear that file abstraction seems to be a powerful concept to be implemented for large files that

compress well and are likely to be browsed several times before they are actually retrieved. Images are a good example of this.

6 Concluding Remarks

In this paper, we have developed an approximate closed queuing network model of the Unitree mass storage system used at NASA's Center for Computational Sciences. The approximations were validated by discrete even simulation and the complete model was calibrated and validated with measurements taken at the NASA Goddard Space Flight Center Unitree Mass Storage System. The model was then used to analyze the performance of the mass storage system with respect to interactive requests for retrieving and storing files. Eight classes were used as the workload, in order to accurately represent the different file sizes present in the get and put requests. The model was then used to study the performance of the system for various hit ratios at the disk cache, to predict the performance as the load increases, and to investigate the use of compression and file abstraction. The major contributions of the paper can be summarized as follows:

- A complete workload characterization of a mass storage system at a large scientific installation (NCCS—NASA's Center for Computational Sciences). The workload characterization showed that most files retrieved are small (1.2 MB), while large files represent a small fraction of the number of retrieved files. The same behavior was observed for storage requests.
- A validated queuing network model that can be used by managers of mass storage system sites to carry out capacity planning studies and support procurement decisions of expensive storage devices, such as tape silos. Workload forecast studies done by the Computer Environments and Research Requirements Committee predict a five-fold increase in workload intensity over the next two years to support NASA's earth and space science research. The model was able to show that, under the predicted workload, the storage time of large files will increase by almost 50 percent and the retrieval time of large files will increase by close to 30 percent.
- A novel and accurate MVA-based approximation for modeling the RAID disks that compose the disk cache at this mass storage system. We proposed a modification to the MVA response time equation and we showed, through simulation, that this modification accurately models the fork-join synchronization present in servicing requests by the striped device.
- Examples on how the complete model could be used to investigate the impact of compression and file abstraction techniques on the performance of mass storage systems. Client compression appears to be more beneficial, since it can offer considerable performance gains with a compression ratio of only 30 percent. Server compression can be beneficial only if the probability that the file is found in the disk cache in compressed form is kept below 25 percent for an average compression ratio of at least 30 percent. The model showed that file abstraction is a powerful concept to be implemented for large files that compress well and are likely to be browsed several times before they are actually retrieved. In fact, gains of the order of 62 percent can be achieved for large files for abstraction factors of 10 percent (not unrealistic for image files).

The model presented in this paper can also be used to assess design alternatives for mass storage systems, as well as to analyze the impact of using devices that incorporate faster technologies at the various levels of the storage hierarchy.

Acknowledgments

We would like to thank Adina Tarshish and Ellen Salmon from NASA's Center for Computational Sciences at Goddard Space Flight Center for assisting us in understanding the Unitree and decrypting its numerous log files, and Jacqueline Lemoigne for explaining to us the wavelet-based compression techniques being developed at NCCS. We would also like to thank Charles Aston, Edward Bender, and Brian Christianson for providing us with documentation and answering questions regarding the CONVEX server's hardware architecture and the striped file system. The work of Daniel Menascé was partially supported by CESDIS. Finally, we would like to thank the referees for their thoughtful comments and suggestions, which improved the quality of the paper.

References

[1] R.H. Katz, T.E. Anderson, J.K. Ousterhout, and D.A. Patterson, "Robo-Line Storage: Low Latency, High Capacity Storage Systems over Geographically Distributed Networks," Technical Report UCB/S2K-91-3, Univ. of California, Berkeley, Mar. 1994.

[2] E.L. Miller and R.H. Katz, "An Analysis of File Migration in a Unix Supercomputing Environment," Technical Report UCB/CSD-92-712, Univ. of California, Berkeley, Mar. 1993.

[3] A. Tarshish and E. Salmon, "The Growth of the Unitree Mass Storage System at the NASA Center for Computational Sciences," *Proc. Third NASA GSFC Conf. Mass Storage Systems and Technologies*, pp. 19-21, College Park, Md., Oct. 1993.

[4] S. Coleman and S. Miller, "Mass Storage System Reference Model: Version 4," *Proc. Goddard Conf. Mass Storage Systems and Technologies*, College-Park, Md., 1992.

[5] K.K. Ramakrishnan and J.S. Emer, "Performance Analysis of Mass Storage Service Alternatives for Distributed Systems," *IEEE Trans. Software Eng.*, vol. 15, no. 2, pp. 120-133, Feb. 1989.

[6] E. Drakopoulos and M.J. Merges, "Performance Study of Client-Server Storage Systems," *Proc. 11th IEEE Symp. Mass Storage Systems*, pp. 67-72, Monterey, Calif., 1991.

[7] E. Drakopoulos and M.J. Merges, "Performance Analysis of Client-Server Storage Systems," *IEEE Trans. Computers*, vol. 41, no. 11, pp. 1,442-1,452, Nov. 1992.

[8] O.I. Pentakalos and Y. Yesha, "Evaluating the Effect of Online Data Compression on a Disk Cache of a Mass Storage System," *Proc. Fourth Goddard Conf. Mass Storage Systems and Technologies*, College Park, Md., Mar. 1995.

[9] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance Reliable Secondary Storage," *ACM Computing Surveys*, vol. 36, no. 3, pp. 145-185, Aug. 1994.

[10] D.A. Menascé, V.A.F. Almeida, and L.W. Dowdy, *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*. Englewood Cliffs, N.J.: Prentice Hall, 1994.

[11] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data*. New York: John Wiley & Sons, 1990.

[12] H. Schwetman, "CSIM: A C-based, Process Oriented Simulation Language," *Proc. 1991 Winter Simulation Conf.*, pp. 387-396, 1991.

[13] S. Chen and D. Towsley, "A Performance Evaluation of RAID Architectures," Technical Report UM-CS-1992-067, Univ. of Massachusetts, Amherst, 1992.
[14] E.K. Lee and R.H. Katz, "An Analytic Performance Model of Disk Arrays," *Proc. 1993 SIGMETRICS Conf. Measurement and Modeling of Computer Systems*, pp. 98-109, Santa Clara, Calif., 1993.
[15] R.H. Johnson, "UNIX Metrics: Is the Data in Open Systems the Same from Platform to Platform?" *Proc. 1994 Computer Measurement Group Conf.*, pp. 796-807, Orlando, Fla., 1994.
[16] J. Bleker, "UNIX Capacity Planning Hints and Tips," *Proc. 1994 Computer Measurement Group Conf.*, pp. 1-7, Orlando, Fla., 1994.
[17] S. Sarawagi and M. Stonebraker, "Single Query Optimization for Tertiary Memory," Technical Report UCB:S2K-94-45, Univ. of California, Berkeley, 1994.
[18] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice Hall, 1982.
[19] J.C. Lowery, "Calibration and Predictive Modeling of Computer Systems," PhD thesis, Vanderbilt Univ., Nashville, Tenn., 1992.
[20] J.C. Lowery and L.W. Dowdy, "New Directions in Model Calibration," *Proc. CMG '91 Int'l Conf. Management and Performance Evaluation of Computer Systems*, Dec. 1991.
[21] The Computer Environments and Research Requirements Committee, "NASA Earth and Space Sciences Computing Requirements for 1997-2004," internal report, Jan. 1995.
[22] O.I. Pentakalos and Y. Yesha, "Online Data Compression for Mass Storage File Systems," Technical Report TR-CS-95-05, Univ. of Maryland Baltimore County, July 1995.
[23] D. Mack, "compress version 4.0," Usenet comp.sources.misc:volume20 /compress.
[24] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Information Theory*, vol. 23, no. 3, pp. 337-343, 1977.
[25] J.C. Tilton, M. Manohar, and J.A. Newcomer, "Earth Science Data Compression Issues and Activities," *Remote Sensing Reviews*, vol. 9, pp. 271-298, Sept. 1994.

Odysseas I. Pentakalos received his PhD in computer science from the University of Maryland Baltimore County in 1996. He was then awarded a U.S. National Research Council fellowship and spent a year at the NASA Goddard Space Flight Center where he continued his research on the performance modeling of mass storage systems. Dr. Pentakalos is currently a visiting assistant professor at the University of Maryland Baltimore County. He is the recipient of a 1995 Graduate Students Researchers Program Fellowship from NASA. His research interests include mass storage systems, performance evaluation and modeling of computer systems, operating systems, databases, and distributed systems. Dr. Pentakalos is a member of the IEEE, the IEEE Computer Society, and the ACM.

Daniel A. Menascé received his PhD in computer science from the University of California at Los Angeles in 1978. He is a professor of computer science at George Mason University (GMU) and the associate director of the Center for the New Engineer at GMU. He is also a special member of the Graduate School at the University of Maryland Baltimore County. Prior to joining GMU in 1992, he was a faculty member at the Pontificia Universidade Catolica do Rio de Janeiro from 1978 to 1992, where he was department chair from 1981 to 1983. He was the president of the Brazilian Computer Society from 1987 to 1989. He has published three books and more than 90 refereed papers in journals and conference proceedings. His research interests are performance modeling of computer systems, parallel computing, and distributed systems. He is a fellow of the ACM, a member of the IEEE and the IEEE Computer Society, and a member of the Computer Measurement Group.

Milton Halem received his PhD in mathematics from the Courant Institute of Mathematical Sciences, New York University, in 1968. He is currently the chief of the Space Data and Computing Division at NASA's Goddard Space Flight Center (GSFC) in Greenbelt, Maryland. In 1977, he went to the Goddard Space Flight Center (GSFC) in Washington, DC, where he served as project scientist on the Global Atmospheric Research Program (GARP) and as the head of the Global Modeling and Simulation Branch. Prior to that, he was a senior mathematical analyst with the Goddard Institute for Space Studies (GISS) in New York. Since joining NASA, Dr. Halem has been instrumental in bringing supercomputers to GSFC, most recently, the CRAY Y-MP, along with a mass data storage and delivery system that comprises one of the world's most powerful scientific data computing complexes. He is the recipient of numerous awards, including the "Goddard Exceptional Performance Award" in 1974 and 1978; the "NASA Medal for Exceptional Scientific Achievement" in 1979; the "NASA Certificate for Outstanding Performance" in 1982, and the NASA Medal for Outstanding Leadership in 1990. His research interests include simulation studies of the impact of space observing systems on weather forecasting and climate modeling. He is a member of the IEEE and the IEEE Computer Society.

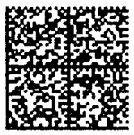
Yelena Yesha received her PhD in computer science from Ohio State University in 1988. In addition to serving as the director of CESDIS, she holds a joint appointment as a professor in the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County, and at the University of Maryland's Institute for Advanced Computer Studies in College Park. During 1994, she served as the director of the Center for Applied Information Technology at the National Institute of Standards and Technology. Dr. Yesha is a senior member of the IEEE, a member of the IEEE Computer Society, a member of the ACM, and a member of the New York Academy of Sciences. She has served as a member of President Clinton's Electronic Commerce Acquisition Team and as a member of the Task Force on National Information Infrastructure Services. Dr. Yesha has authored or coauthored 16 refereed journal articles, 23 conference papers, and a book chapter. Her research interests include data management, digital libraries, distributed systems, and distributed databases.

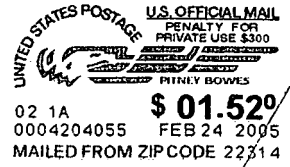
Organization TC2800 Bldg./Room

U. S. DEPARTMENT OF COMMERCE
COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450
IF UNDELIVERABLE RETURN IN TEN DAYS

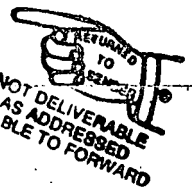
OFFICIAL BUSINESS

Jeff





AN EQUAL OPPORTUNITY EMPLOYER



7590

TEJPAL S. HANSRA
REGISTRATION NO. 38,172
3705 CANTERBURY LANE, #6
[redacted]